

Standards

Standards Translation Rules for CBPR+ Credit Transfers and Cash Management

CBPR+ Translation Rules Descriptions

The Standards Translation Rules describe how to translate source MT or MX messages to their equivalent target MX or MT messages. This document is the Translation Rule Descriptions for the Standards Translation Rules for CBPR+ Credit Transfer and Cash Management Messages.

30 May 2023

[Preface 5](#_Toc136351212)

[1 Introduction 8](#_Toc136351213)

[2 Basic Translation Function Descriptions 9](#_Toc136351214)

[3 MT to MX Translation Rule Descriptions 20](#_Toc136351215)

[3.1 Customer Party Translation Rule Descriptions 20](#_Toc136351216)

[3.1.1 MT\_To\_ MXAnyBIC 20](#_Toc136351217)

[3.1.2 MT\_To\_ MXPartyAccount 21](#_Toc136351218)

[3.1.3 MT\_To\_ MXAccount35 23](#_Toc136351219)

[3.1.4 MT\_To\_ MXFATFIdentification 24](#_Toc136351220)

[3.1.5 MT\_To\_ MXFATFNameAndAddress 31](#_Toc136351221)

[3.1.6 MT\_To\_ MXPartyNameAndAddress 42](#_Toc136351222)

[3.1.7 MT\_To\_MXPartyNameAndStructuredAddress 44](#_Toc136351223)

[3.1.8 MT\_To\_MXUltimateParty 47](#_Toc136351224)

[3.2 Financial Institution Translation Rule Descriptions 55](#_Toc136351225)

[3.2.1 MT\_To\_MXBICFI 55](#_Toc136351226)

[3.2.2 MT\_To\_MXClearingIdentifier 56](#_Toc136351227)

[3.2.3 IsMTClearingSystemCodeInList 60](#_Toc136351228)

[3.2.4 MT\_To\_MXClearingSystemToNameAndAddressLine 62](#_Toc136351229)

[3.2.5 MT\_To\_MXFinancialInstitutionNameAndUnstructuredAddress 63](#_Toc136351230)

[3.2.6 MT72INS\_To\_MXAgent 66](#_Toc136351231)

[3.2.7 MT\_To\_MXFinancialInstitutionAccount 70](#_Toc136351232)

[3.2.8 MT\_To\_MXIntermediaryAgent 73](#_Toc136351233)

[3.3 Other Translation Rule Descriptions 76](#_Toc136351234)

[3.3.1 MT\_To\_MXRate 76](#_Toc136351235)

[3.3.2 MT\_To\_MXDate 76](#_Toc136351236)

[3.3.3 MT\_To\_MXTime 77](#_Toc136351237)

[3.3.4 MT\_To\_MXOffset 79](#_Toc136351238)

[3.3.5 MT\_To\_MXTimeOffset 80](#_Toc136351239)

[3.3.6 MT\_To\_MXDateTimeOffset 81](#_Toc136351240)

[3.3.7 MT\_To\_MXCurrencyAmount 82](#_Toc136351241)

[3.3.8 MT70ROC\_To\_MX35Text 83](#_Toc136351242)

[3.3.9 MT\_To\_MXRemittanceInformation 84](#_Toc136351243)

[3.3.10 MT72\_To\_MXText 86](#_Toc136351244)

[3.3.11 MT\_To\_ MXRegulatoryReporting 87](#_Toc136351245)

[3.3.12 MT\_To\_ MXInstruction 90](#_Toc136351246)

[3.3.13 MTDate4\_To\_MXDate 92](#_Toc136351247)

[3.3.14 MT\_To\_MXSum 93](#_Toc136351248)

[3.3.15 MT\_To\_MX InstructionForCreditorAgent 94](#_Toc136351249)

[3.3.16 MT\_To\_MX InstructionForCreditorAgent2 98](#_Toc136351250)

[3.3.17 MT\_To\_MXBNF\_TSU\_RI 105](#_Toc136351251)

[3.3.18 MT\_To\_MXField72NewCodeWords 106](#_Toc136351252)

[3.3.19 MT\_To\_MX\_Serial\_CoverScenario 112](#_Toc136351253)

[3.3.20 MT\_To\_MXReturn72 113](#_Toc136351254)

[3.3.21 MT\_To\_ MXRegulatoryReporting2 117](#_Toc136351255)

[3.3.22 MT\_To\_MXReject72 118](#_Toc136351256)

[3.3.23 MT\_To\_MXPurpose 122](#_Toc136351257)

[3.3.24 MT\_To\_MXRINewCodeWords 124](#_Toc136351258)

[3.3.25 MT\_To\_MXField79 127](#_Toc136351259)

[3.3.26 MT\_To\_MXField76 132](#_Toc136351260)

[3.3.27 MT\_To\_MXField79\_2 137](#_Toc136351261)

[4 MX to MT Translation Rule Descriptions 140](#_Toc136351262)

[4.1 Customer PartyTranslation Rule Description 140](#_Toc136351263)

[4.1.1 MX\_To\_MTAnyBIC 140](#_Toc136351264)

[4.1.2 MX\_To\_MTFATFIdentification 141](#_Toc136351265)

[4.1.3 MX\_To\_MTFATFNameAndAddress 153](#_Toc136351266)

[4.1.4 MX\_To\_MTFATFNameAndAddress2 159](#_Toc136351267)

[4.1.5 MX\_To\_MTFATFID\_CUST\_NIDN 165](#_Toc136351268)

[4.1.6 MX\_To\_MTUltimateParty 171](#_Toc136351269)

[4.1.7 MX\_To\_MTBirthInformation 173](#_Toc136351270)

[4.1.8 MX\_To\_MTPartyNameAndUnstructuredAddress 174](#_Toc136351271)

[4.1.9 MX\_To\_MTPartyNameAndStructuredAddress 176](#_Toc136351272)

[4.1.10 MX\_To\_MTPartyNameAndAddressLEI2 186](#_Toc136351273)

[4.1.11 MX\_To\_MTPartyNameAndAddressLEI1 191](#_Toc136351274)

[4.1.12 MX\_To\_MTPartyNameAndStructuredAddressNoNumber 192](#_Toc136351275)

[4.2 Financial Institution Translation Rule 193](#_Toc136351276)

[4.2.1 MX\_To\_MTBICFI 193](#_Toc136351277)

[4.2.2 MX\_To\_MTClearingIdentifier 194](#_Toc136351278)

[4.2.3 MX\_To\_MTClearingIdentifierAndChannel 197](#_Toc136351279)

[4.2.4 MX\_To\_MTClearingIdentifierChoice 201](#_Toc136351280)

[4.2.5 MX\_To\_MTNameAndAddressToClearingSystemIdentifier 202](#_Toc136351281)

[4.2.6 MX\_To\_MTAgent 203](#_Toc136351282)

[4.2.7 MX\_To\_MTBICNameAgent 205](#_Toc136351283)

[4.2.8 MX\_To\_MTFinancialInstitutionNameAndUnstructuredAddress 206](#_Toc136351284)

[4.2.9 MX\_To\_MTFinancialInstitutionNameAndStructuredAddress 209](#_Toc136351285)

[4.2.10 MX\_To\_MT54A 217](#_Toc136351286)

[4.2.11 MX\_To\_MT53A 218](#_Toc136351287)

[4.2.12 MX\_To\_MTAgentGeneric 221](#_Toc136351288)

[4.3 Other Translation Rule Descriptions 230](#_Toc136351289)

[4.3.1 MX\_To\_MTAccount 230](#_Toc136351290)

[4.3.2 MX\_To\_MTSettlementAccount 231](#_Toc136351291)

[4.3.3 MX\_To\_MTDate 232](#_Toc136351292)

[4.3.4 MX\_To\_MTTimeOffset 233](#_Toc136351293)

[4.3.5 MX\_To\_MTDateTimeOffset 234](#_Toc136351294)

[4.3.6 MX\_To\_MTRate 235](#_Toc136351295)

[4.3.7 MX\_To\_MTCurrencyAmount 236](#_Toc136351296)

[4.3.8 MX\_To\_MTRemittanceInformation 238](#_Toc136351297)

[4.3.9 MXText\_To\_MT72 253](#_Toc136351298)

[4.3.10 MX\_To\_MT72FullField 255](#_Toc136351299)

[4.3.11 AppendComplexMT72 266](#_Toc136351300)

[4.3.12 MX\_To\_MT23E 269](#_Toc136351301)

[4.3.13 MX\_To\_MTCharSet 274](#_Toc136351302)

[4.3.14 MX\_To\_MT72FullField2 276](#_Toc136351303)

[4.3.15 MX\_To\_MTAddressLineType 305](#_Toc136351304)

[4.3.16 MX\_To\_MTStartingLineCharacter 308](#_Toc136351305)

[4.3.17 MX\_To\_MTRemittanceInformation2 310](#_Toc136351306)

[4.3.18 MX\_To\_MT72FullField3 313](#_Toc136351307)

[4.3.19 MX\_To\_MTRegulatoryReporting 320](#_Toc136351308)

[4.3.20 MX\_To\_MT79FullField 325](#_Toc136351309)

[4.3.21 MX\_To\_MT72RETN 328](#_Toc136351310)

[4.3.22 MX\_To\_MT72Or79REJT 336](#_Toc136351311)

[4.3.23 MX\_To\_MTEmptyLine 342](#_Toc136351312)

[4.3.24 MX\_To\_MT79CANC 344](#_Toc136351313)

[4.3.25 MX\_To\_MT76RCANC 346](#_Toc136351314)

[4.3.26 MX\_To\_MT23E\_BIS 349](#_Toc136351315)

[4.3.27 MX\_To\_MTPartyNameAndStructuredAddress1 353](#_Toc136351316)

[4.3.28 MX\_To\_MT79CANC2 357](#_Toc136351317)

[4.3.29 MX\_To\_MT75CANC 360](#_Toc136351318)

[4.3.30 MX\_To\_MT76CANC 361](#_Toc136351319)

[5 Annex 363](#_Toc136351320)

[6 Change Log 365](#_Toc136351321)

[Legal Notices 374](#_Toc136351322)

# Preface

**About this document**

This document provides the descriptions for the *Standards Translation Rules for CBPR+ Credit Transfer and Cash Management Messages*. Some translation descriptions are for basic translation functions, common to all Standards translations. Other translation descriptions are specific functions for a dedicated pair of messages. Translation rules cover both directions MT to MX and MX to MT directions.

Credit Transfer translation rules and Cash Management translation rules cover the Standards Release MT 2021 version and the latest MX release at the date of publication of this document.

This document forms part of the following Translation Rules document set:

* CBPR+ Credit Transfer and Cash Management Messages Usage Guidelines published on [MyStandards](https://www2.swift.com/mystandards/#/c/cbpr/landing)
* Translation Rules (per message pair) described in excel documents
* Translation Rule Descriptions (for CBPR+ Credit Transfer and Cash Management Messages)
* Error code Master List document

All four documents must be used together to obtain an understanding of how to translate source MT or MX messages to their equivalent target MX or MT messages.

The starting point to understand the translation is the excel document linked to a pair of messages. For example, pacs.008 to MT103 translation. The source message elements are listed and the translation rules applied to the elements are defined in the excel document. If the rule is too complex or uses a translation function that can be reused in other message translation pairs, then the “excel rules” may refer to a function with a name starting with MX\_To\_MT (or MT\_To\_MX) which is described in this word document. The “Find” function can be used to locate the function in this document. So purpose of this document is not to be read sequentially but to refer to it to find the details of a function mentioned in the excel documents. Complex functions in this document can also refer to other more “granular” functions also described here.

In order to better understand how the translation rules work, it is recommended to use the [Translation Portal](https://www2.swift.com/mystandards/#/c/cbpr/landing) for testing purpose but also for documentation purpose as the translation rules are also available from the tool. The readers can then create their own examples with data commonly used by their organisation and verify how they are translated. Please note that the first time you want to access the Portal, a request must be submitted.

**Audience**

This document is for the following SWIFT audiences who are interested in understanding how to translate MT or MX messages:

* SWIFT internal
* SWIFT users (community)
* Vendors part of the SWIFT Partner Programme

**Version**

These offline document versions of CBPR+ translations published on the CBPR+ Translation Portal are released periodically after completion of planned phases of CBPR+ definition. Planned dates of publication are available on the [CBPR+ MyStandards group](https://www2.swift.com/mystandards/#/c/cbpr/landing). The date is used to differentiate the versions.

A change log is used in each document (excel and word documents) to explain the changes between versions.

In addition, significant additions compared to the previous version are marked in “track change mode”. Significant deletions compared to the previous version are marked in strikethrough. Typographical error corrections are not marked.

A Release Note is also published with each release listing the changes versus the previous release. The Relelase Note is published separately [on MyStandards](https://www2.swift.com/mapping/#/).

**Terminology**

This document contains terms that are consistent with SWIFT terminology and documentation on FIN message types (including ISO 15022 Messages) and ISO 20022 messages. SWIFT defines these terms in the SWIFT Glossary.

New terminology in this document includes:

**Semantic unit**: The smallest entity in a message that still has a semantic or individual meaning within the context of that message.

**Source message:** The input message being used as the source or starting point of the translation.

**Target message:** The output message being used as the target or ending point for the translation.

**Document conventions**

This document uses the following typographical conventions and abbreviated forms of product and service titles:

|  |  |
| --- | --- |
| **Bold** | Commands |
| *Italics* | File, directory, and document names. |
| Courier | Examples and formal descriptions. |
| MT | Message Type (FIN) |
| MX | Message Type (XML) |

**Related documentation**

* *SWIFT User Handbook, Standards MT*, relevant *Message Reference Guides*
* *ISO 20022 Message Definition Report - Payments Standards - Clearing and Settlement* and related XML schemas and instances
* *ISO 20022 Message Definition Report - Payments Standards - Initiation* and related XML schemas and instances

The latest version of the *SWIFT Message Reference Guides* is available at [www.swift.com](http://www.swift.com/) > Support > Documentation (User Handbook).

The latest version of the *ISO 20022 documentation* is published on the [www.iso20022.org](http://www.iso20022.org/) website.

SWIFT Standards Intellectual Property Rights (IPR) Policy - End-User License Agreement

This document is provided as part of the SWIFT Translation Rules that are licensed subject to the terms and conditions of the *SWIFT Translation Rules IPR Policy - End-User License Agreement*, available at [www.swift.com > About Us > Legal > IPR Policies > SWIFT Translation Rules IPR Policy.](https://www.swift.com/about-us/legal/ipr-policies/translation-ipr-policy?tl=en#topic-tabs-menu)

# 1 Introduction

The *Standards CBPR+ Translation Rules* describe how to translate source MT or CBPR+ MX messages to their equivalent target CBPR+ MX or MT messages.

Latest and up to date CBPR+ usage guidelines and translation rules are available on the [CBPR+ MyStandards group](https://www2.swift.com/mystandards/#/c/cbpr/landing). This document is provided for convenience and offline reference only. It is updated periodically after completion of planned phases of CBPR+ definition. Planned dates of publication are available on the [CBPR+ MyStandards group.](https://www2.swift.com/mystandards/#/c/cbpr/landing)

This document expresses no warranty of performance of translation rules, and SWIFT offers no support for this offline documentation.

This document is provided as part of the SWIFT Translation Rules that are licensed subject to the terms and conditions of the *SWIFT Translation Rules IPR Policy - End-User License Agreement*, available at [www.swift.com > About Us > Legal > IPR Policies > SWIFT Translation Rules IPR Policy.](https://www.swift.com/about-us/legal/ipr-policies/translation-ipr-policy?tl=en#topic-tabs-menu)

This Translation Rule Descriptions document contains:

|  |  |
| --- | --- |
| Section 1  Introduction | Provides an introduction to this document. |
| Section 2  Basic Translation Function Descriptions | Describes basic translation functions. |
| Section 3  MT to MX Translation Rule Descriptions | Describes MT to MX translation rules. |
| Section 4  MX to MT Translation Rule Descriptions | Describes MX to MT translation rules. |
| Section 5  Translation Additional Information | Describes additional information related to translation rules |
| Section 6  Change Log | Describes the changes between document versions |

# 2 Basic Translation Function Descriptions

Basic translation functions are the set of functions used throughout all the Standards translations. This section, Basic Translation Function Descriptions, describes these basic translation functions in terms of their definition and formats.

|  |  |
| --- | --- |
| **Function name** | **Description** |
| **AppendTo**  **(Value, Target)** | Appends the source value to the target node. The source can be a node, a string, an integer, a real number or a date and its type must correspond to the type of the target node. |
| **AppendToNextLine (Value, Target)** | Appends the source value to the first available line in the target node. The source can be a node, a string, an integer, a real number or a date and its type must correspond to the type of the target node. If the target node is empty, the source value is written starting on the first line.  If the target node is not empty, the function adds a Carriage Return Line Feed (CRLF) to the end of the content already present in the target prior to writing the source value on the next available line.  This function is used with a target node composed of several lines for examples 4\*35x. |
| **ReturnFirstLineEmpty(Target,n)** | Returns the number of the first line empty in a structure of n lines \*35 characters separated with CRLF. If no line found empty, returns 0. If the first 2 lines are not empty, and the next one is empty, returns 3 |
| **ReturnFirstLineEmptyExtended(Target,n,LineLength)** | Returns the number of the first line empty in a structure of n lines \* x characters (LineLength) separated with CRLF. If no line found empty, returns 0. If the first 2 lines are not empty, and the next one is empty, returns 3 |
| **Concatenate (Value1,**  **Value2, Value3, … ValueN)** | Returns a string by concatenation of Value1, Value2,… ValueN respecting the order of occurrence of the Value parameters. |
| **Copy**  **[(Value)]** | Copies the source value in the target node. The source can be a node, a string, an integer, a real number or a date and its type must correspond to the type of the target node. The target is given by the target path. Formats:   * **Copy:** Copies the value from the source path to the target path. * **Copy(Value):** Copies the explicit source value to the target path. * **Copy("boolean value"):** Copies the value "true" or “false" to a boolean variable of the target path. * **Copy("value"):** Copies the value between “” to the target path eg. a code. Note that if the target path is developed up to the code, the function indicates the value of the last node in the target path. The function gives the information from a type perspective. * **Copy(SubfieldValue):** Copies a part (that is a subfield) of a field from an MT message. For example, Field 97B is defined by (Qualifier) (Data Source Scheme) (Account Type Code) (Account Number). Copy(AccountNumber) means copy the part of the field that contains the account number.   If the value to be copied is returned by a translation function, then Copy is implicit in the spreadsheet. |

|  |  |
| --- | --- |
| **DecimalSignificantDigits (Value)** | Returns the number of significant decimal digits in a source value that is a decimal number. For example, for a source value 0120.4560000, the function returns 3 as number of significant decimal digits as the source decimal part is indeed equivalent to .456. |
| **DeleteChar**  **(Value, FromPosition,**  **ToPosition)** | Returns the result after deletion of characters in the source value starting from the FromPosition (included) to the ToPosition (included). Positions are integers. |
| **DeletePattern**  **(Value, Pattern)** | Returns the result after deletion of the string indicated by Pattern in the source value. If the pattern is not found, then the value is returned unchanged. Formats:   * **DeletePattern(Value, "Pattern"):** In this format, the pattern is given as a string between quotes. For example   DeletePattern(Date, "-") will remove the "-" in the date string parameter.   * **DeletePattern(Value, Pattern):** In this format, the pattern is generic and the function will search for the instances of the pattern. For example, YYYY-MM-DD is a date pattern. The function will look for date instances formatted as YYYY-MMDD and delete them.   Note that a combination of a string pattern and a generic pattern should also be supported by the function. For example, DeletePattern(DateTime, "T"HH:MM:SS) with DateTime "200712-08T23:12:01" will return “2007-12-08” |
| **EqualTo**  **(also used "=")** | Logical operator that checks whether two values are equal, that is whether the value on the left of the operator is the same as the value on the right |
| **EquivalentCode**  **(Value, SourceCodeList,**  **TargetCodeList)** | Returns the code from the TargetCodeList that is equivalent to the code from the SourceCodeList that corresponds with the source value. If no equivalent code is found, an empty string is returned. It is part of the specifications to provide the  SourceCodeList, the TargetCodeList and the code equivalences between the two lists. |
| **ExtractBetweenPattern**  **(Value, FromPattern,**  **{ToPattern1, ToPattern2, …**  **ToPatternN})** | Extracts a string from the source value between the  FromPattern and the first occurring ToPattern from the list. The FromPattern and ToPattern are not part of the output string.  If the source value contains more than one occurrence of the FromPattern, then the function extracts between the first occurring FromPattern up to the first occurring ToPattern from the list. If the source value does not contain the FromPattern, then an empty string is returned.  If the source value does not contain (any of) the ToPattern(s), then the function extracts starting at the FromPattern (not included) until the end of the source value.  If the list contains only ToPattern1, then the curly brackets "{" and “}” are omitted.  If the FromPattern is equal to the ToPattern, then the function will extract between two occurrences of the same pattern for example ExtractBetweenPattern(Value, "/", "/") will extract the |

|  |  |
| --- | --- |
|  | string between the first occurring slash and the second slash.  If the information to be extracted is spread over more than one line, then the function deletes the Carriage Return Line Feed (CRLF) used to separate consecutive lines in an MT field with multiple lines and concatenates the content on the different lines in the returned value. Formats:   * **ExtractBetweenPattern(Value, "FromPattern",**   **{"ToPattern1", "ToPattern2", … "ToPatternN"}):** In this format, the patterns are given as strings between quotes. For example, ExtractBetweenPattern(Value, “/ROC/”, {“/INV/”, “/IPI/”, “/RFB/”})   * **ExtractBetweenPattern(Value, FromPattern, {ToPattern1, ToPattern2, … ToPatternN}):** In this format, the patterns are generic and the function will search in the source value for the instances of the pattern. For example, YYYY-MM-DD is a date pattern. The function will look for date instances formatted as YYYY-MM-DD.   Note that a combination of a string pattern and a generic pattern should also be supported by the function. For Example,  ExtractBetweenPattern(DateTime, YYYY”-“, "T"HH:MM:SS) with  DateTime "2007-12-08T23:12:01" will return "12-08” |
| **ExtractFromPattern**  **(Value, [FromPosition],**  **FromPattern)** | Extracts a string starting at the FromPattern until the end of the source value. If the (optional) FromPosition is present, then the function will search for the FromPattern starting at the position in the source value.  For example:  ExtractFromPattern(ABC/DEF/GHI, "/") returns the string DEF/GHI.  ExtractFromPattern(ABC/DEF/GHI, 5, "/") returns the string GHI.  The returned string does not include the FromPattern.  If the source value does not contain the FromPattern, then an empty string is returned. If the source value contains more than one occurrence of the FromPattern, then the function extracts between the first occurring FromPattern until the end of the source value.  If the information to be extracted is spread over more than one line, then the function deletes the Carriage Return Line Feed (CRLF) used to separate consecutive lines in an MT field with multiple lines and concatenates the content on the different lines in the returned value. If the information to be extracted is limited to one line, then the function does not return the CRLF in the output string. Formats:   * **ExtractFromPattern(Value, [FromPosition],**   **"FromPattern"):** In this format, the pattern is given as strings between quotes. For example, ExtractFromPattern(Value, “/ROC/”)   * **ExtractFromPattern(Value, [FromPosition],**   **FromPattern):** In this format, the pattern is generic and the function will return the string following the generic pattern. |

|  |  |
| --- | --- |
|  | For example, ExtractFromPattern(DateTime, YYYY-MM-DD) with DateTime “2007-12-08T23:12:01” returns “T23:12:01” Note that a combination of a string pattern and a generic pattern should also be supported by the function. For example, ExtractFromPattern(DateTime, YYYY-MM-DD“T”) with  DateTime “2007-12-08T23:12:01” returns “23:12:01” |
| **ExtractLines**  **(Value, FromPattern,**  **[ContinuationPattern])** | Extracts the lines from a string with multiple lines beginning at the first line that starts with the specified "FromPattern" and continuing on the next lines that start with the  "ContinuationPattern". The extraction stops as soon as a line is encountered that does not start with the specified  ContinuationPattern. If the optional ContinuationPattern is not given, then the extraction starts from the FromPattern until the end of line. When no lines start with the specified "FromPattern", an empty string is returned. The FromPattern is included in the output string.  If the information to be extracted is spread over more than one line, then the function deletes the Carriage Return Line Feed (CRLF) used to separate consecutive lines in an MT field with multiple lines. If the length of a line < 35 characters, the ContinuationPattern on the next line is replaced by a space “ ” prior to concatenating t with the content of the next line. If the length of a line = 35, the ContinationPattern on the next line is deleted and is NOT replaced by a space “ “ prior to concatenating with the content of the next line. If the information to be extracted is limited to one line, then the function does not return the CRLF in the output string.  Format:  • **ExtractLines(Value, "FromPattern",**  **["ContinuationPattern"]):** The patterns are strings given between quotes. |
| **ExtractLinesAsIs**  **(Value, FromPattern,**  **[ContinuationPattern])** | The function works as **ExtractLines** function described above but the ContinuationPattern is NOT replaced by a space prior to concatenating the content on the different lines in the returned value. |
| **ExtractPattern (Value, Pattern)** | Extracts a string in the source value that is of the pattern format. If the source value contains more than one occurrence of the pattern, then the first occurrence is returned.  If the pattern is not found, then an empty string is returned. If the information to be extracted is spread over more than one line, then the function deletes the Carriage Return Line Feed (CRLF) used to separate consecutive lines in an MT field with multiple lines prior to concatenating the content on the different lines in the returned value. **Formats:**   * **ExtractPattern(Value, “Pattern”:** In this format, the function returns the first occurrence of the exact pattern as given between quotes. For example,   ExtractPattern(CurrencyAmount, “USD”) will extract the currency “USD” from a CurrencyAmount “USD1500,”   * **ExtractPattern(Value, Pattern):** In this format, the function returns the first occurrence of the generic pattern. For example, ExtractPattern(DateTime, HH:MM:SS) will extract the time "23:23:32" from a DateTime "2007-12-01T23:23:32" Note that a combination of a string pattern and a generic pattern should also be supported by the function. For example, ExtractPattern(DateTime, "T"HH:MM:SS) with DateTime "200712-08T23:12:01" will return " T23:12:01” |

|  |  |
| --- | --- |
| **ExtractTillPattern**  **(Value, [FromPosition],**  **TillPattern)** | Extracts a string in the source value until the TillPattern. If the (optional) FromPosition is present, then the function will return the string in the source value starting at the FromPosition until the TillPattern.  For example:  ExtractTillPattern(ABC/DEF/GHI, "/") returns the string ABC ExtractTillPattern(ABC/DEF/GHI, 5, "/") returns the string DEF The returned string does not include the TillPattern.  If the source value does not contain the TillPattern, then the source value (or source value starting at FromPosition) is returned. If this source value is limited to one line, then the function does not return the CRLF in the output string.If the source value contains more than on occurrence of the  TillPattern, then the function extracts until the first occurrence of the TillPattern.  If the information to be extracted is spread over more than one line, then the function deletes the Carriage Return Line Feed (CRLF) used to separate consecutive lines in an MT field with multiple lines prior to concatenating the content on the different lines in the returned value. Formats:   * **ExtractTillPattern(Value, [FromPosition], "TillPattern"):** In this format, the pattern is given as strings between quotes. For example, ExtractTillPattern(Value, “/”) * **ExtractTillPattern(Value, [FromPosition], TillPattern):** In this format, the pattern is generic and the function will return the string up to the generic pattern. For example, ExtractTillPattern(DateTime, HH:MM:SS) with DateTime   “2007-12-08T23:12:01” returns “2007-12-08T”  Note that a combination of a string pattern and a generic pattern should also be supported by the function. For example, ExtractTillPattern(DateTime, “T”HH:MM:SS) with DateTime  “2007-12-08T23:12:01” returns “2007-12-08” |
| **DeleteExtractLines(Value, FromPattern,**  **[ContinuationPattern])** | Delete from Value the information extracted with the function ExtractLines |
| **DeleteLine(Element,LineNumber)** | In a multiline element, delete the line with LineNumber and return the element with line removed and replaced by the next ones if any. |
| **FirstOccurrenceOf (Element)** | Returns the first occurrence of an element that may have many occurrences. For example, an address line may have many lines.  Another equivalent notation is Element[1]. |
| **GreaterEqualTo (also used "**≥**")** | Logical operator that checks whether the value on the left of the operator is equal to or greater than the value on the right. |
| **GreaterThan**  **(also used ">")** | Logical operator that checks whether the value on the left of the operator is greater than the value on the right. |
| **Ignore** | Ignore means there is no translation for the semantic unit. |
| **IsAbsent (Value)** | Checks whether the source value is absent or empty. It returns a boolean value (True or False). |
| **IsAbsentPattern (Value, Pattern)** | Checks whether the pattern is absent in the source value. It returns a boolean value (True or False). Formats:  • **IsAbsentPattern(Value, "Pattern"):** In this format, the |

|  |  |
| --- | --- |
|  | pattern is given as a string between quotes. If Value is "This is a Test", IsAbsentPattern(Value, "T") returns "False".  • **IsAbsentPattern(Value, Pattern):** In this format, the pattern is generic and the function will search for the first instance of the pattern. For example, with the pattern HH:MM:SS, the function will look for a time instance formatted as HH:MM:SS. If value is "This is an example", the function returns "True"  Note that a combination of a string pattern and a generic pattern should also be supported by the function. For Example, IsAbsentPattern(DateTime, "T"HH:MM:SS) with DateTime "2007-12-08T23:12:01" will return "False". |
| **IsAccount(Value)** | Checks whether Value is formatted as an account based on the below criteria. It returns a Boolean value (True or False).  IF IsAbsent(Value) THEN Return False  ELSEIFValue starts with “//CH” THEN Return True  ELSEIF Value does not start with “//” Return True  ELSE Return False. |
| **IsAlphabetic (Value)** | Checks whether all characters of the source value are alphabetical. It returns a boolean value (True or False). |
| **IsAlphabeticOrNumeric (Value)** | Checks whether all characters of the source value are alphabetical or numerical. It returns a boolean value (True or False). |
| **IsBEI (Value)** | Checks whether the source value is a valid BEI and returns a boolean value (True or False). |
| **IsBIC (Value)** | Checks whether the source value is a valid BIC and returns a boolean value (True or False). |
| **IsCountryCode (Value)** | Checks whether the source value is a valid country code. It returns a boolean value (True or False). |
| **IsEmpty (Value)** | Checks whether the source value is an empty string. It returns a boolean value (True or False). |
| **IsIBAN (Value)** | Checks whether the source value is a valid IBAN. It returns a boolean value (True or False).  At a minimum, the function should perform the mod-97 check as described below.  An IBAN is validated by converting it into an integer and performing a basic mod-97 operation (as described in ISO 7064) on it. If the IBAN is valid, the remainder equals 1.  The algorithm use for IBAN validation is as follows.   1. The account must be made of 2 alphabetic characters followed by 2 numeric characters followed by minimum 1 and maximum 30 alphanumeric characters. The first 2 alphabetic characters must be a valid country code (as per ISO Standards 3166). 2. Move the four initial characters to the end of the string 3. Replace each letter in the string with two digits, thereby expanding the string, where A = 10, B = 11, ..., Z = 35 4. Interpret the string as a decimal integer and compute the remainder of that number on division by 97   If the remainder is 1, the check digit test is passed and the IBAN might be valid.  Example (fictitious United Kingdom bank, sort code 12-34-56, account number 98765432):   * IBAN:  GB82 WEST 1234 5698 7654 32 * Rearrange:   W E S T12345698765432 G B82 * Convert to integer:  3214282912345698765432161182 * Compute remainder:  3214282912345698765432161182 mod 97 = 1   The ISO20022 IBAN pattern must be met as well:  [A-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30} |
| **IsPresent**  **(Element)** | Checks whether an element is present and not empty. The element may be a structure, part of a structure or a simple element. It returns a boolean value (True or False). |
| **IsPresentPattern**  **(Value, Pattern)** | Checks whether the pattern is present in the source value. It returns a boolean value (True or False). Formats:   * **IsPresentPattern(Value, "Pattern"):** In this format, the pattern is given as a string between quotes. If Value is "2007-12-08T23:12:01", IsPresentPattern(Value, "T") returns "True". * **IsPresentPattern(Value, Pattern):** In this format, the pattern is generic and the function will search for the first instance of the pattern. For example, with the pattern HH:MM:SS, the function will look for a time instance formatted as HH:MM:SS. If value is "2007-12-08T23:12:01", the function returns "True" Note that a combination of a string pattern and a generic pattern should also be supported by the function. For Example, IsPresentPattern(DateTime, "T"HH:MM:SS) with DateTime   "2007-12-08T23:12:01" will return "True" |
| **Length (Value )** | Returns the length of the source value. If the source value is an empty string, zero is returned. |

|  |  |
| --- | --- |
| **LessEqualTo (also used "**≤**")** | Logical operator that checks whether the value on the left of the operator is less than or equal to the value on the right. |
| **LessThan**  **(also used "<")** | Logical operator that checks whether the value on the left of the operator is less than the value on the right. |
| **NumberOfOccurrences (Element)** | Returns the number of occurrences of the source element. The element can be a simple element or a more complex structure. |
| **OccurrenceOf (Element)** | Returns the occurrence number (the occurrence position) of the specified repetitive source element that can be a complex structure. |
| **PadLeft**  **(Value, Character, Length)** | Returns the result after padding the value to the left with the specified character until a value of the specified length is obtained. If the length of the value is greater than or equal to the specified length, no padding characters are added. |
| **ReplacePattern**  **(Value, Pattern1, Pattern2)** | Returns the result after replacing Pattern1 with Pattern2 in the source value.  If pattern1 is not found, the source value is returned unchanged.  Formats:   * **ReplacePattern(Value, "Pattern1", "Pattern2"):** In this format, the pattern parameters are given as a string between quotes. For example ReplacePattern(DateTime, "T", "") will remove "T" in the Value "2007-12-08T23:12:01". * **ReplacePattern(Value, Pattern1, Pattern2):** In this format, the pattern parameters are generic. All instances of pattern1 will be formatted with pattern2 in Value. For example, if pattern1 is HH:MM:SS and pattern2 is HHMM, the function will look for all time instances formatted as HH:MM:SS and replace them by a time formatted as HHMM. If value is "2007-12-08T23:12:01", the returned string is "2007-1208T2312".   Note that a combination of a string pattern and a generic pattern should also be supported by the function. For Example, ReplacePattern(DateTime, "T"HH:MM:SS, HHMMSS) with DateTime "2007-12-08T23:12:01" will return "2007-1208231201". |
| **SecondOccurrenceOf (Element)** | Returns the second occurrence of an element that may have many occurrences. For example, an address line may have many lines.  Another equivalent notation is Element[2]. |
| **SignificantDigits (Value)** | Returns the number of significant digits in a source value that is  a (decimal) number. For example, for a source value  0120.4560000, the function returns 6 as number of significant digits as the source value is indeed equivalent to 120.456 |
| **SplitInLines**  **(Value, NumberOf**  **Characters,**  **[ContinuationPattern])** | Returns the result of splitting the source value in lines of the specified number of characters.  If the source value is split over more than one line, then the function adds the Carriage Return Line Feed (CRLF) used to separate consecutive lines in an MT field with multiple lines at the end of each line, with exception of the last line.  In the same case where the source value is split over more than |
|  | one line, the optional ContinuationPattern -if present- is written at the start of each line following the first line. |
| **StartWith ("Value")** | Checks whether the value on the left of the function starts with the pattern and returns a boolean value (True or False). |
| **Stop translation** | Stops translation of the message. |
| **Substring**  **(Value, From**  **Position, [Length])** | Returns a substring from the source value starting at the  FromPosition (included) and of the indicated Length. If no Length parameter is present or if the length of the source value is less than the Length parameter, then a substring is returned starting at the FromPosition until the end of the source value. |
| **SumOfOccurrences (Element)** | Returns the sum of all the element source values. Element must be an integer or decimal. |
| **TrimLeft**  **(Value, Character)** | Returns the result after removal of the specified character starting from the extreme left of the value until a character different from the character parameter is encountered. |
| **TrimRight**  **(Value, Character)** | Returns the result after removal of the specified character starting from the extreme right of the value until a character different from the character parameter is encountered. |
| **Truncate**  **(Value, Position )** | Returns the result after truncation of the source value after the position parameter. |
| **WithinList**  **(Value, CodeList)** | Checks whether the source value is present in the code list and  returns a boolean value (True or False). It is part of the specifications to provide the code list. |

# 

# 3 MT to MX Translation Rule Descriptions

This section provides translation rule descriptions for the translation rules for MT to MX Credit Transfer and Cash Management messages. Descriptions are grouped as follows:

* Customer Party Translation Rule Descriptions
* Financial Institution Translation Rule Descriptions
* Other Translation Rule Descriptions

## 3.1 Customer Party Translation Rule Descriptions

The translation rules described in this section are for customer party translation rules.

### 3.1.1 MT\_To\_ MXAnyBIC

**Name**

MT\_To\_MXAnyBIC

**Business description**

The function translates an MT Any BIC as a party identification by writing the code in the appropriate MX party identification element.

**Format**

**MT\_To\_MXAnyBIC** (MTAnyBIC ; MXParty)

**Input**

MTAnyBIC: Any BIC identifying a party in an MT message.

**Output**

MXParty: the result of this function is immediately put into the correct MX element inside the target component typed *PartyIdentification135*.

**Preconditions** None.

**Formal description**

MXParty.Identification.OrganisationIdentification.AnyBIC = MTAnyBIC

**Example 1 MT Source:**

:50A:GEBABEBB (Ordering Customer bank)

**MX Translation:**

<Dbtr>

<Id>

<OrgId>

<AnyBIC>GEBABEBB</AnyBIC>

</OrgId>

</Id>

</Dbtr>

**Example 2 MT Source:**

:59A:FOMOUS33 (Beneficiary Customer)

**MX Translation:**

<Cdtr>

<Id>

<OrgId>

<AnyBIC>FOMOUS33</AnyBIC>

</OrgId>

</Id>

</Cdtr>

### 3.1.2 MT\_To\_ MXPartyAccount

**Name**

MT\_To\_MXPartyAccount

**Business description**

The function translates an MT party account to an MX account by deleting the slash “/”and writing the remainder in the appropriate MX account element in Identification/Other/Identification or in Identification/IBAN.If the MT party account starts with “//CH” then the number following this code (the CHIPS Universal Identifier) will be written in the MX account element, complemented by a code “CUID” that will be written in the account scheme element.

**Format**

**MT\_To\_MXPartyAccount**(MTAccount ; MXAccount)

**Input**

MTAccount: party account in the MT format (/34x).

**Output**

MXAccount: the result of this function -an MX account- is immediately put into the correct MX element inside the target component typed *CashAccount38*.

**Preconditions** None.

**Formal description**

/\*Check whether the account starts with //CH. If it starts with

//CH the CHIPS Universal Identifier following this (6 digits) will be written in the account identification. The code “CUID” is written in the scheme name element to indicate that the account identification is a CHIPS Universal Identifier\*/

/\* Local Variables

Account : string \*/

**IF** Substring(MTAccount,1,4) = “//CH” THEN

Account = **Substring**(MTAccount,5)

**IF Length** (Account) > 0 THEN

Identification.Other.SchemeName.Code = “CUID”

Identification.Other.Identification = Account

**ELSE**

**T11013**

**ENDIF /\* CH case \*/**

**ELSE**

/\*Drop the slash. Account is a local variable\*/

Account = **Substring**(MTAccount, 2)

**IF Length**(Account) > 0 **THEN**

**IF** **IsIBAN**(Account)

Identification.IBAN = Account

**ELSE**

Identification.Other.Identification = Account  **ENDIF**

**ENDIF**

**ENDIF**

**Example 1 MT Source:**

:50K:*Vanmusten Ilya* /123456789

*Brugmannlaan 415*

*BE/Brussels*

*(Italic not illustrated by this function)*

**MX Translation:**

<DbtrAcct>

<Id>

<Othr>

<Id>123456789</Id>

</Othr>

</Id>

</DbtrAcct>

**Example 2 MT Source:**

:59:/BE16001216377774

*Vanmusten Ilya*

*Brugmannlaan 415*

*BE/Brussels*

*(Italic not illustrated by this function)*

**MX Translation:**

<CdtrAcct>

<Id>

<IBAN>BE16001216377774</IBAN>

</Id>

</CdtrAcct>

**Example 3 MT Source:**

:59://CH123456

*Vanmusten Ilya*

*Brugmannlaan 415*

*BE/Brussels*

*(Italic not illustrated by this function)*

**MX Translation:**

<CdtrAcct>

<Id>

<Othr>

<Id>123456</Id>

<SchmeNm>

<Cd>CUID</Cd>

</SchmeNm>

</Othr>

</Id>

</CdtrAcct>

### 3.1.3 MT\_To\_ MXAccount35

**Name**

MT\_To\_MXAccount35

**Business description**

The function translates an MT account to an MX account if the MT account length is less than 35 characters in which case it is translated to MX Account/Identification/Other/Identification or to Identification/IBAN. If the MTaccount starts with “//CH” then the number following this code (the CHIPS Universal Identifier) will be written in the MX account element, complemented by a code “CUID” that will be written in the account scheme element.

IF the MT account length is equal to 35 characters, an error is generated during the translation and the account number is truncated with “+” added to indicate truncation was needed.

**Format**

**MT\_To\_MXAccount35**(MTAccount ; MXAccount)

**Input**

MTAccount: account in the MT format (35x).

**Output**

MXAccount: the result of this function -an MX account- is immediately put into the correct MX element inside the target component typed *CashAccount39*.

**Preconditions** None.

**Formal description**

/\*Check whether the account starts with //CH. If it starts with

//CH the CHIPS Universal Identifier following this (6 digits) will be written in the account identification. The code “CUID” is written in the scheme name element to indicate that the account identification is a CHIPS Universal Identifier\*/

**IF** Substring(MTAccount,1,4) = “//CH” THEN

MXAccount.Identification.Other.SchemeName.Code = “CUID”

MXAccount.Identification.Other.Identification =

**Substring**(MTAccount,5)

**ELSE**

**IF** **Length**(MTAccount)> 34 THEN

MXAccount.Identification.Other.Identification = Concatenate(Substring(MTAccount,1,33),”+”)

~~T20050~~ T0000T /\* Error code described in Error code list \*/

**ELSE**

**IF** **IsIBAN**(Account)

MXAccount.Identification.IBAN = MTAccount

**ELSE**

MXAccount.Identification.Other.Identification = MTAccount

**ENDIF**

**ENDIF**

**ENDIF**

### 3.1.4 MT\_To\_ MXFATFIdentification

**Name**

MT\_To\_MXFATFIdentification

**Business description**

Subfield 1 of an MT field 50F identifies the Ordering Customer in line with FATF Special Recommendation VII by providing either an account number or a unique identification of the customer. This function translates the unique identifications used in the MT to their equivalent MX FATF compliant identifications by:

* checking whether the identification is continued in Subfield 2 of field 50F (use of code “8” in Subfield 2). If so, the identification is completed prior to translation.
* isolating the code in the identification to determine the MX target element and write the actual identification in the appropriate element.
* splitting the country or -depending on the type of identification- country with issuer from the identification and write this information in the MX Issuer element.
* If line 8 in NameAndAddress is the continuation of PartyIdentifier and if the concatenation of Identifier and Line 8 is greater than 35 character, line 8 will be truncated. Truncation is indicated with a sign “+” at the end of the data.
* If the MTPartyIdentifier is generated from a previous translation MX to MT from OrganisationID with a SchemeNameCode in the external ISO list but different from CUST, TXID or EMPL then MTPartyIdentifier translation back to OrgID is handled in this function. The expected structure is therefore **CUST**/CountryCode/OrgSchemeNameCode[ MXIssuer]/ID where “SPACE MXIssuer” is optional (see MX\_To\_MTFATFIdentification).
* If the MTPartyIdentifier is generated from a previous translation MX to MT from PrivateID with a SchemeNameCode in the external ISO list but different from CUST, TXID, EMPL, ARNU, CCPT, DRLC, NIDN, SOSE then MTPartyIdentifier translation back to PrivateID is handled in this function. The expected structure is therefore **CUST**/CountryCode/PrivSchemeNameCode[ MXIssuer]/ID where “SPACE MXIssuer” is optional (see MX\_To\_MTFATFIdentification).
* Note that if the MX SchemeNameCode is TXID, EMPL or CUST, after translation to MTPartyIdentifier, it is not possible to identify whether it came from Organisation ID or Private ID. It is translated then to Private ID.
* Current possible limitation: If the MTPartyIdentifier is generated from a previous translation MX to MT with a SchemeNameCode in the external ISO list but with no MT equivalent code, and if the SchemeNameCode is present in both lists Private and Organisation Scheme Name Code, it is not possible to identify if MT PartyIdentifier has to be translated back to PrivateID or Organisation ID. By implementation, it will be translated back to Organisation ID. This comment is to cope with possible future evolution of the ISO external code lists for SchemeName Code

**Format**

**MT\_To\_MXFATFIdentification**(MTPartyIdentifier, MTNameAndAddress ; MXFATFId) **Input**

MTPartyIdentifier: Subfield 1 of field 50F carrying an FATF compliant customer identification

MTNameAndAddress: Subfield 2 of field 50F carrying an FATF compliant customer name and address or identifications in line with the format description of field 50F (date and place of birth, customer number or national identity number).

**Output**

MXFATFId: the result of this function is directly put into the correct MX element inside the target component typed *PartyIdentification135*.

**Preconditions**

This function is used if the Party Identifier is NOT an account ie., if

Substring(MTPartyIdentifier, 1, 1) NOT = “/”

**Formal description**

/\* Local variables

MXOrgSchemeCode, MXPrivSchemeCode, MTIssuer, MTCode8, MTFullId, MTFATFCode, MTFATFID, MTCountry, MTId : string

FlagOrgId, FlagPrivId : Boolean \*/

FlagOrgId = “false”

FlagPrivID = “false”

/\*Check whether the MTPartyIdentifier is continued in MTNameAndAddress on a line starting with “8”\*/

/\*Basic function *ExtractLines* extracts the continued information. “8/” must not be repeated so the function is used without continuation string to limit the extraction to a maximum of one line. If “8/” is not found an empty string is returned. MTCode8 is a local variable\*/

MTCode8 = **ExtractLines**(MTNameAndAddress, “8/”)

/\*First assess whether continuation under “8/” is related to Subfield 1 or Subfield 2 by checking whether the first character is a slash (case where the MTPartyIdentifier carries an account number, not to be continued under “8/”. Otherwise assess whether “8/” is the continuation of the

MTPartyIdentifier by checking whether the MTPartyIdentifier reaches the maximum length. In this case the MTPartyIdentifier will be combined with extract MTCode8 that might be empty. Substring is taken to delete “8/”. MTFullId is a local variable If both cases are negative, “8/” is the continuation of a number “6/” or “7/” in Subfield 2 and its translation is handled by the *MT\_To\_MXFATFNameAndAddress* function\*/

MTFullId= MTPartyIdentifier

**IF Length**(MTPartyIdentifier) = 35 THEN

/\* Line 8 is the continuation of PartyIdentifier defined with FATFIdentification \*/

MTFullId = **Append**(**Substring**(MTCode8, 3), MTFullId)

**ENDIF**

/\*Split code from identification in MTFullId. MTFATFCode and

MTFATFId are local variables\*/

MTFATFCode = **Substring**(MTFullId, 1, 4)

MTFATFId = **Substring**(MTFullId, 6)

/\*Split country or country and issuer from the identification.

MTCountry, MTIssuer and MTId are local variables\*/

**IF** MTFATFCode = “CUST” OR “EMPL” OR “DRLC”

/\*Assumption is that the source message is correct.

Therefore CUST, EMPL or DRLC should be in line with field 50F format specifications: the code must be followed by a slash, the country code, a slash, the issuer of the identification, a slash and the identification. The scenario also takes into account the possible structure CountryCode//Number where issuer is missing (see MX to MT translation) \*/

/\*Isolate country and issuer from identification. Restrict identification to maximum 35 characters\*/

MTCountry = **Substring**(MTFATFId, 1, 2)

/\* check if the Issuer is present and follows by “/” and by Identification. This is not validated by the network. So check is reinforced in the translation \*/

**IF** **IsPresentPattern**(Substring(MTFATFId, 4), “/”)THEN

**{** MTIssuer = **ExtractTillPattern**(Substring(MTFATFId, 4), “/”)

MTId = **ExtractFromPattern**(Substring(MTFATFId, 4), “/”)

**IF** MTId IsEmpty THEN

/\* case where there is no info after the slash. Then MT Id and MT issuer will have the same content as no possibility to identify which information is present \*/

MTId = “NOTPROVIDED”

**ENDIF**

/\* If MT issuer is generated from OrgID.Other.SchemeName.Code with a value different from CUST, TXID or EMPL then a structure like “CUST/BE/GS1G MXIssuer/ID” is expected (see MX\_To\_MTFATFIdentification) and should therefore be translated back to OrgID and not to PrivateID. So the code after CUST/BE/ is checked versus the external ISO list ExternalOrganisationIdentification1Code \*/

**IF** MTFATFCode = “CUST” **AND** **WithinList**(**Substring**(MTIssuer,1,4),ExternalOrganisationIdentification1Code)AND **Substring**(MTIssuer,1,4)is NOT in List {CUST, TXID, EMPL} THEN

MXOrgSchemeCode = **Substring**(MTIssuer,1,4)

MTIssuer = **Substring**(MTIssuer,6)

/\* a space is present per construction between MXOrgSchemeCode and the remaining issuer information if present \*/

FlagOrgId = “true”

/\* If MT issuer is generated from PrivID.Other.SchemeName.Code with a value that has no MT equivalent then a structure like “CUST/BE/MXCode MXIssuer/ID” is expected (see MX\_To\_MTFATFIdentification) and should therefore be translated back to PrivID. So the code after CUST/BE/ is checked versus the external ISO list PersonIdentification1Code \*/

**ELSEIF** MTFATFCode = “CUST” **AND** **WithinList**(**Substring**(MTIssuer,1,4),ExternalPersonIdentification1Code)AND **Substring**(MTIssuer,1,4)is NOT in List {CUST, TXID, EMPL,ARNU,CCPT,DRLC,NIDN,SOSE} THEN

MXPrivSchemeCode = **Substring**(MTIssuer,1,4)

MTIssuer = **Substring**(MTIssuer,6)

/\* a space is present per construction between MXOrgSchemeCode and the remaining issuer information if present \*/

FlagPrivId = “true”

**ENDIF**

**}**

**ELSE**

**/\* IF** NOT **IsPresentPattern**(Substring(MTFATFId, 4), “/”) \*/

MTId = **Substring**(MTFATId, 4)

MTIssuer = “”

**ENDIF**

/\* MT ID contains the Identifier and the MTCode8 information if any \*/

IF **Length**(MTId) > 35 Then

MTId = Concatenate(Substring(MTId,1,34), “+”)

ENDIF

**ELSE**

/\* MTFATFCode is not in the list {EMPL, CUST, DRLC} \*/

/\*Assumption is that the source message is correct. For all other codes, field 50F format specification requires that the code is followed by a slash, the country code, a slash and the identification\*/

/\*Isolate country from identification. Restrict identification to maximum 35 characters\*/

MTCountry = **Substring**(MTFATFId, 1, 2)

MTId = **Substring**(MTFATFId, 4)

IF MTId **IsEmpty** THEN

MTId = “NOTPROVIDED”

ENDIF

IF **Length**(MTId) > 35 Then

MTId = **Concatenate**(**Substring**(MTId, 1, 34), “+”)

ENDIF

**ENDIF**

/\*Translation to target\*/

**IF** MTFATFCode = “ARNU” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code = “ARNU”

**ELSEIF** MTFATFCode = “CCPT” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= “CCPT”

**ELSEIF** MTFATFCode = “CUST” THEN

**IF** FlagOrgId = “false” AND FlagPrivId = “false”THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

=”CUST”

**ELSIF** FlagOrgID = “true” THEN

Identification.OrganisationIdentification.Other.Identification = MTId

Identification.OrganisationIdentification.Other.SchemeName.Code

= MXOrgSchemeCode

**ELSEIF** FlagPrivID = “true” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= MXPrivSchemeCode

**ENDIF**

**ELSEIF** MTFATFCode = “DRLC” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= “DRLC”

**ELSEIF** MTFATFCode = “EMPL” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= “EMPL”

**ELSEIF** MTFATFCode = “NIDN” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= “NIDN”

**ELSEIF** MTFATFCode = “SOSE” THEN

Identification.PrivateIdentification.Other.Identification = MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= “SOSE”

**ELSEIF** MTFATFCode = “TXID” THEN

Identification.PrivateIdentification..Other.Identification

= MTId

Identification.PrivateIdentification.Other.SchemeName.Code

= “TXID”

**ENDIF**

/\*Write country and issuer or country in the same instance of the target PrivateIdentification component\*/

**IF** MTFATFCode = “CUST” OR “EMPL” OR “DRLC” AND **IsPresent**(MTIssuer) THEN

/\* MTIssuer is not empty \*/

/\*Concatenate country with issuer No truncation is needed as Country and Issuer is a subset of PartyIdentifier which has a max length of 35 characters. \*/

**IF** FlagOrgId = “false” THEN

Identification.PrivateIdentification.Other.Issuer =

**Concatenate**(MTCountry, “/”, MTIssuer)

**ELSE**

Identification.OrganisationIdentification.Other.Issuer =

**Concatenate**(MTCountry, “/”, MTIssuer)

**ENDIF**

**ELSE**

/\*Fill in the Issuer with the Country Code\*/

**IF** FlagOrgId = “false” THEN

Identification.PrivateIdentification.Other.Issuer = MTCountry

**ELSE**

Identification.OrganisationIdentification.Other.Issuer = MTCountry

**ENDIF**

**ENDIF**

### 3.1.5 MT\_To\_ MXFATFNameAndAddress

**Name**

MT\_To\_MXFATFNameAndAddress

**Business description**

Subfield 2 of an MT field 50F identifies the Ordering Customer in line with FATF Special Recommendation VII by combining:

* name and address
* name and date and place of birth
* name and national identity number or LEI
* name and customer number

Every line in Subfield 2 of field 50F starts with a number indicating the type of information provided. This function translates the structured information in the MT to MX FATF compliant equivalents by:

* + checking the numbers to determine the MX target element
  + checking whether numbers are repeated if allowed by the format description on the following line. If so, the information following the number will be completed prior to the translation
  + If LEI is present in 6/ with the following pattern 6/CountryCode/LEIC/LEIIdentifier (20 char) then it is translated to OrganisationIdentification/LEI.. LEI is translated only if the MTPartyIdentifier is an account or has a structure resulting from a previous translation MX to MT from Organisation ID/ Other ID (example, “CUST/CountryCode/GS1G..”). If the MTPartyIdentifier starts with a code as defined in 50F UHB, then it is translated to Private ID and therefore incompatible with the translation of LEI to Organisation ID
  + If subfield 1 is an account, 8/ will never be the continuation of an account.
  + If subfield 1 is NOT an account and length(subfield1) = 35 char then 8/ is the continuation of subfield 1. This is handled in function MT\_To\_MXFATFIdentification.
  + If (subfield 1 is not an account and length < 35 char OR subfield 1 is an account) then search if 8/ is the continuation of 6/ else search if 8/ is the continuation of 7/.
  + If line 8 is present and cannot be identified as the continuation of the PartyIdentifier or continuation of Line 6 or line 7, information is disregarded.

**Format**

**MT\_To\_MXFATFNameAndAddress**(MTPartyIdentifier, MTNameAndAddress ; MXFATFId)

**Input**

MTPartyIdentifier: Subfield 1 of field 50F carrying an account number or other FATF compliant customer identification

MTNameAndAddress: Subfield 2 of field 50F carrying an FATF compliant customer name and address or identifications in line with the format description of field 50F (date and place of birth, customer number or national identity number).

**Output**

MXFATFId: the result of this function is immediately put into the correct MX element inside the target component typed *PartyIdentification135*.

**Preconditions**

None.

**Formal description**

/\*MTNameAndAddress is defined by 4 lines of 35 characters.

MTNameAndAddress[1] indicates the first line\*/

/\*Translation of the name present on line(s) starting with “1/” \*/

/\*Basic function *ExtractLines* extracts the name. The complete name could be present on several lines so the function is used with continuation string “1/”. Substring is taken to delete “1/” in the output string \*/

As the same structure is used for Line Number “1/”, “2/” and “3/” for Field 50F and 59F, a common function is used MT\_to\_MXPartyNameAndStructuredAddress. This function includes also the extraction of 3/CountryCode/LEIC/ which will only occur for the Creditor 59F, not for the Debtor 50F. LEI for the Debtor is extracted from 6/ and the case is handled below \*/

**MT\_To\_MXPartyNameAndStructuredAddress**(MTNameAndAddress; MXFATFId)

/\* Check if Number 6/ is present with LEI. In that case Organisation ID will be filled and not Private ID, meaning that 4/ birth date and 5/ place of birth and 7/NIDN will be ignored as well as 8/ as 8/ will never be the continuation of LEI line \*/

**IF** **Length(ExtractPattern**(MTNameAndAddress, “6/”)) > 0 THEN

MTCode6 = **Substring(ExtractLines**(MTNameAndAddress,

“6/”), 3)

/\* 2 cases : either the information following 6/ is a LEI (case 1) or is a customer ID (case 2). Case 2 is handled later in the function \*/

/\* IF LEI, then the line 6/ will never be completed as max char is 28 after deletion of “6/” due to the pattern “6/CountryCode/LEIC/LEIIdentifier exact 20 char.

LEI is translated only if the MTPartyIdentifier is an account or has a structure as defined below, signature it results from a previous translation MX to MT from Organisation ID/ Other ID. If the MTPartyIdentifier starts with a code as defined in 50F UHB, then it is translated to Private ID and therefore incompatible with the translation of LEI to Organisation ID. \*/

**IF** Substring(MTcode6,1,2) IsCountry AND Substring(MTcode6,3,6) = “/LEIC/” AND Substring(MTcode6,9)has pattern [A-Z0-9]{18,18}[0-9]{2,2} AND

**{**Substring(MTPartyIdentifier,1,1) = “/”

/\* MTPartyIdentifier is an account \*/

OR

/\* MTPartyIdentifier has a structure ***CUST****/CountryCode/OrgSchemeNameCode[ MXIssuer]/ID where “SPACE MXIssuer” is optional (for the construction, see MX\_To\_MTFATFIdentification). For example, “CUST/US/GS1G IssuerName/Identification”. As this information is translated back to Organisation Identification, LEI is translated while the other numbers 4/, 5/, 6/ and 7/ are not translated to Private Identification as Organisation ID and Private ID are exclusive. \*/*

**[IF Substring**(MTPartyIdentifier, 1,4) = “CUST” AND

**IsCountryCode** (**Substring**(MTPartyIdentifier,6,2))AND

**WithinList(Substring**(MTPartyIdentifier,9,4), ExternalOrganisationIdentification1Code) AND

**Substring**(MTPartyIdentifier,9,4)NOT In List {CUST,TXID,EMPL}

**]**

/\* ExternalOrganisationIdentification1Code is defined in ISO list of external code sets \*/

**}**

THEN

Identification.OrganisationIdentification.LEI = Substring(MTcode6,9)

/\*If organisation Id is filled Private Id cannot be filled as elements are exclusive \*/

**EXIT function**

**ENDIF**

**ENDIF**

/\*The section below describes the translation of date and place of birth (lines starting with “4/” and “5/”), customer number (lines starting with “6/”) or national identity number (lines starting with “7/”). Date and place of birth will translate to the PrivateIdentification.DateAndPlaceOfBirth.BirthDate component. Customer number and/or national identity number will

translate to different instances of the PrivateIdentification.Other component\*/

/\*Translation of date and place of birth present on lines starting with “4/” and “5/”. As per field 50F format specifications,”4/” must not be used without “5/” and vice versa\*/

**IF Length(ExtractPattern**(MTNameAndAddress, “4/”)) > 0

/\*Basic function ExtractLines extracts date and place of birth. “4/” and “5/” must not be repeated so the function is used without continuation stringMTDate and MTPlace are local variables. Substrings are taken to delete “4/” and

“5/” in the output strings\*/

MTDate = **Substring(ExtractLines**(MTNameAndAddress,“4/”), 3)

MTPlace = **Substring(ExtractLines**(MTNameAndAddress,“5/”), 3)

/\*Translated MTDate to the MX format and write in the occurrence of the target PrivateIdentification component\*/

Identification.PrivateIdentification.DateAndPlaceOfBirth.BirthDate = **MT\_To\_MXDate**(MTDate)

/\*Write country and city in the same occurrence of PrivateIdentification component as for the date\*/

Identification.PrivateIdentification.DateAndPlaceOfBirth.CountryOfBirth = **Substring**(MTPlace, 1, 2)

Identification.PrivateIdentification.DateAndPlaceOfBirth.Ci tyOfBirth = **Substring**(MTPlace, 4)

**ENDIF**

/\*Translation of customer number present on a line starting with “6/”\*/

/\*Check whether Subfield 1 of field 50F (MTPartyIdentifier) carries a customer number. If this is the case, a line starting

with “6/” is ignored as the customer number in the

MTPartyIdentifier will be translated\*/

**IF** **Substring**(MTPartyIdentifier, 1, 4) NOT = “CUST”

/\*Check whether “6/” is present \*/

**IF** **Length(ExtractPattern**(MTNameAndAddress, “6/”)) > 0

/\*Basic function *ExtractLines* extracts the customer details. “6/” must not be repeated so the function is used without continuation string. MTCode6 is a local variable.

Substring is taken to delete “6/” in the output string\*/

MTCode6 = **Substring(ExtractLines**(MTNameAndAddress,“6/”), 3)

/\*Check whether in MTNameAndAddress a line starting with “8/” is present\*/

/\*Basic function *ExtractLines* extracts the continued information. “8/” must not be repeated. If “8/” is not found an empty string is returned. MTCode8 is a local variable\*/

MTCode8 = **Substring**(**ExtractLines**(MTNameAndAddress, “8/”), 3)

/\*If “8/” is present, first assess whether continuation under “8/” is related to Subfield 1 or Subfield 2 by checking whether the first character is a slash, ie.,case where the MTPartyIdentifier carries an account number, not to be continued under “8/”. Otherwise assess whether “8/” is the continuation of the MTPartyIdentifier by checking whether the MTPartyIdentifier reaches the maximum length.

If this is not the case, check if MTCode6 reaches the maximum length. If this is the case MTCode6 will be combined with MTCode8 after taking a substring to delete “8/”. MTFullId is a local variable. \*/

MTFullId = MTCode6

**IF** ((**Substring(**(MTPartyIdentifier,1,1) NOT = “/” AND

**Length**(MT PartyIdentifier) NOT = 35) OR **Substring**(MTPartyIdentifier,1,1) = “/” )

**AND Length**(MTCode6) = 33 **AND** **Length**(MTCode8)>0

/\* Line 8 is not the continuation of PartyIdentifier but the continuation of Line 6\*/

THEN MTFullId = Concatenate(MTCode6, MTCode8)

**ENDIF**

/\*Assumption is that the source message is correct. Therefore a customer number should be in line with field 50F format specifications: “6/” must be followed by the country code, a slash, the issuer of the number, a slash and the customer identification number. The case where Issuer is absent is also handled in order to make the translation smoother. MTIssuer, MTCountry and MTNum are local variables\*/

MTCountry = **Substring**(MTFullId, 1, 2)

/\* check if the Issuer is present and follows by “/” and by Identification. This is not validated by the network. So check is reinforced in the translation \*/

**IF** **IsPresent**(Substring(MTFullId, 4), “/”)THEN

MTIssuer = **ExtractTillPattern**(Substring(MTFullId, 4), “/”)

MTNum = **ExtractFromPattern**(Substring(MTFullId, 4), “/”)

**IF** MTNum IsEmpty THEN

/\* case where there is no info after the slash. Then MT Num and MT issuer will have the same content as no possibility to identify which information is present \*/

MTNum = “NOTPROVIDED”

**ENDIF**

**ELSE**

MTNum = **Substring**(MTFullId, 4)

MTIssuer = “”

**ENDIF**

/\* MTNum contains Customer ID and possibly MTCode8 \*/

/\*Write number and issuer in a next occurrence of the target PrivateIdentification.Other component\*/

/\* Check the length of MTNum \*/

**IF** **Length**(MTNum)> 35 THEN

Identification.PrivateIdentification.Other.Identification = **Concatenate**(**Substring**(MTNum,1,34),”+”))

**ELSE**

Identification.PrivateIdentification.Other.Identification = MTNum

**ENDIF**

Identification.PrivateIdentification.Other.Scheme Name.Code = “CUST”

**ISPresent**(MTIssuer) THEN

/\* MTIssuer is not empty \*/

/\*Concatenate country with issuer No truncation is needed as Country and Issuer is a subset of PartyIdentifier which has a max length of 35 characters. \*/

Identification.PrivateIdentification.Other.Issuer =

**Concatenate**(MTCountry, “/”, MTIssuer)

**ELSE**

/\*Fill in the Issuer with the Country Code\*/

Identification.PrivateIdentification.Other.Issuer = MTCountry

**ENDIF**

**ENDIF** /\* EndIf Length(ExtractPattern(MTNameAndAddress, “6/”)) > 0 \*/

**ENDIF** /\* Endif Substring(MTPartyIdentifier, 1, 4) NOT = “CUST” \*/

/\*Translation of national identity number present on a line starting with “7/”\*/

/\*Check whether Subfield 1 of field 50F (MTPartyIdentifier) carries a national identity number. If this is the case, a line starting with “7/” is ignored as the identity number in the MTPartyIdentifier will be translated\*/

**IF** **Substring**(MTPartyIdentifier, 1, 4) NOT = “NIDN”

/\*Check whether “7/” is present\*/

**IF Length(ExtractPattern**(MTNameAndAddress, “7/”)) > 0

/\* the “7/” Information can be copied only if there is still one occurrence of PrivateIdentification.Other left as CBPR+ UG limits to 2 \*/

**IF NumberOfFreeOccurrences**(MXFATFID.Identification.PrivateIdentification.Other)GreaterOrEqual to 1 THEN

**{**

/\*Basic function *ExtractLines* extracts the customer details. “7/” must not be repeated so the function is used without continuation string. MTCode7 is a local variable. Substring is taken to delete “7/” in the output string\*/

MTCode7 = **Substring(ExtractLines**(MTNameAndAddress, “7/”),3)

/\*Check wether in MTNameAndAddress a line starting with 8/ is present\*/

/\*Basic function *ExtractLines* extracts the continued information. “8/” must not be repeated. If “8/” is not found an empty string is returned.

MTCode8 is a local variable\*/

/\*If 8/ is present, first assess whether continuation under “8/” is related to Subfield 1 or Subfield 2 by checking whether the first character is a slash (case where the

MTPartyIdentifier carries an account number, not to be continued under “8/”. Otherwise assess whether “8/” is the continuation of the MTPartyIdentifier by checking whether the MTPartyIdentifier reaches the maximum length. When MTCode6 reaches the maximum length and MT partyIdentifier doesn’t, the MTCode6 will be combined with a substring that deletes the “8/” of MTCode. When neither MTPartyIdentifier nor MTCode6 reach the maximum length and MTCode7 does, a substring of MTCode7 that deletes the country code should be combined with MTCode8 that might be empty. Substring is taken to delete

“8/”. MTFullId is a local variable\*/

MTCode8 = **Substring**(**ExtractLines**(MTNameAndAddress, “8/”),3)

MTFullId = MTCode7

**IF** ((**Substring**(MTPartyIdentifier,1,1) NOT = “/” AND

**Length**(MTPartyIdentifier) NOT = 35) OR **Substring**(MTPartyIdentifier,1,1) = “/”)

AND

**Length**(MTCode6) NOT = 33 AND **Length**(MTCode7) = 33 AND **Length**(MTCode8) > 0

THEN

/\* Line 8 is not the continuation of PartyIdentifier nor the continuation of Line 6 but the continuation of Line 7\*/

MTFullId =**Concatenate**(MTFullId, MTCode8)

**ENDIF**

/\*Assumption is that the source message is correct. Therefore a national identity number should be in line with field 50F format specifications: “7/” must be followed by the country code, a slash and the national identity number\*/

/\*Write the number and the country in a next occurrence of the target PrivateIdentification.Other component \*/

/\* Additional checks added to foresee the case where only the country code is provided like “7/BE” without slash after country code. Although this is not expected as per UHB description, but it would smooth the translation \*/

MTNumber = “”

**IF** **Length**(MTFullId)> 3 AND Substring(MTFullId, 3, 3) = “/” THEN

MTNumber = **Substring**(MTFullId, 4)

**ENDIF**

**IF** MTNumber **IsEmpty** THEN

MTNumber = “NOTPROVIDED”

**ENDIF**

**IF Length**(MTNumber) > 35 THEN

Identification.PrivateIdentification.Other.Identification = **Concatenate**(**Substring**(MTNum,1,34),”+”))

**ELSE**

Identification.PrivateIdentification.Other.Identification = MTNum

**ENDIF**

Identification.PrivateIdentification.Other.SchemeName.Code = “NIDN”

Identification.PrivateIdentification.Issuer = **Substring**(MTFullId,1, 2)

**}**

**ELSE**

**Flag\_MissingInformation = “true”**

**ENDIF /\*** NumberOfFreeOccurrences \*/

**ENDIF**

**ENDIF** /\* PartyIdentifier NOT NIDN \*/

**Example 1 MT Source:**

:50F:*/9854778849*

1/Vanmusten Ilya

2/Brugmannlaan 415

3/BE/Brussels

*(* *Italic not illustrated by thisfunction)*

**MX Translation:**

<Dbtr>

<Nm>Vanmusten Ilya</Nm>

<PstlAdr>

<AdrLine>2/Brugmannlaan 415</AdrLine>

<AdrLine>3/BE/Brussels</AdrLine>

</PstlAdr>

</Dbtr>

**Example 2 MT Source:**

:50F:*/9854778849*

1/Vanmusten Ilya

4/20040830

5/BE/Ukkel

*(* *Italic not illustrated by this function)*

**MX Translation:**

<Dbtr>

<Nm>Vanmusten Ilya</Nm>

<Id>

<PrvtId>

<DtAndPlcOfBirth>

<BirthDt>2004-08-30</BirthDt>

<CityOfBirth>Ukkel</CityOfBirth>

<CtryOfBirth>BE</CtryOfBirth>

</DtAndPlcOfBirth>

</PrvtId>

</Id>

</Dbtr>

**Example 3 MT Source:**

:50F:*/9854778849*

1/Vanmusten Ilya

7/BE/123-456-7890

*(Italic not illustrated by this function)*

**MX Translation:**

<Dbtr>

<Nm>Vanmusten Ilya</Nm>

<Id>

<PrvtId>

<Othr>

<Id>123-456-7890</Id>

<SchmeNm>

<Cd>NIDN</Cd>

</SchmeNm>

<Issr>BE</Issr>

</Othr>

</PrvtId>

</Id>

</Dbtr>

**Example 4 MT Source:**

:50F:*/12584*

1/Name

6/BE/FvW/ABC12345678976543213854847

#### 8/54689

*(Italic not illustrated by this function)*

**MX Translation:**

<Dbtr>

<Nm>Name</Nm>

<Id>

<PrvtId>

<Othr>

<Id>ABC1234567897654321385484754689</Id>

<SchmeNm>

<Cd>CUST</Cd>

</SchmeNm>

<Issr>BE/FvW</Issr>

</Othr>

</PrvtId>

</Id>

</Dbtr>

**Example 5 MT Source:**

:50F:*ARNU/BE/12598766233*

1/Name

6/BE/FvW/ABC1234567

7/BE/1256ABC12345678976543213854847

#### 8/54689

*(Italic not illustrated by this function)*

**MX Translation:**

<Dbtr>

<Nm>Name</Nm>

<Id>

<PrvtId>

*<Othr>*

*<Id>12598766233</Id> <SchmeNm>*

*<Cd>ARNU</Cd> </SchmeNm>*

*<Issr>BE</Issr>*

*</Othr>*

<Othr>

<Id>ABC1234567</Id> <SchmeNm>

<Cd>CUST</Cd>

</SchmeNm>

<Issr>BE/FvW</Issr>

</Othr>

</PrvtId>

</Id>

</Dbtr>

*(Italic not illustrated by this function)*

/\* Line “7/”and “8/” not translated \*/

### 3.1.6 MT\_To\_ MXPartyNameAndAddress

**Name**

MT\_To\_MXPartyNameAndAddress

**Business description**

The function translates an MT party name and address to an MX name and MX address.

**Format**

**MT\_To\_MXPartyNameAndAddress**(MTNameAndAddress ; MXName, MXPostalAddress)

**Input**

MTNameAndAddress: name and address of a party in an MT message.

**Output**

MXName and MXPostalAddress: the result of this function is immediately put into the correct MX element inside the target component typed *PartyIdentification135*.

**Preconditions** None.

**Formal description**

/\*MTNameAndAddress is defined by 4 lines of 35 characters.

MTNameAndAddress[1] indicates the first line not including the Carriage Return Line Feed “*CRLF*” needed as separator between

consecutive lines in an MT field with multiple line format\*/

/\*Translation of the name\*/

Name = MTNameAndAddress[1]

/\*Translation of the address details. MXPostalAddress is a temporary variable. AddressLine in the target is repetitive (max 3 occurrences of 35 char).

PostalAddress.AddressLine[1] indicates the first occurrence etc.\*/

**IF** **IsPresent**(MTNameAndAddress[4])THEN

PostalAddress.AddressLine[1] = MTNameAndAddress[2] PostalAddress.AddressLine[2] = MTNameAndAddress[3]

PostalAddress.AddressLine[3] = MTNameAndAddress[4]

**ELSEIF** **IsPresent**(MTNameAndAddress[3])THEN

PostalAddress.AddressLine[1] = MTNameAndAddress[2]

PostalAddress.AddressLine[2] = MTNameAndAddress[3]

**ELSEIF** **IsPresent**(MTNameAndAddress[2]) THEN

PostalAddress.AddressLine[1] = MTNameAndAddress[2]

**ENDIF**

**Example 1 MT Source:**

:50K:Vanmusten Ilya

Brugmannlaan 415

BE/Brussels/1180

**MX Translation:**

<Dbtr>

<Nm>Vanmusten Ilya</Nm>

<PstlAdr>

<AdrLine>Brugmannlaan 415</AdrLine>

<AdrLine>BE/Brussels/1180</AdrLine>

</PstlAdr>

</Dbtr>

**Example 2 MT Source:**

:50K:Vanmusten Ilya

Brugmannlaan 415

1180 Brussels

Belgium

**MX Translation:**

<Dbtr>

<Nm>Vanmusten Ilya</Nm>

<PstlAdr>

<AdrLine>Brugmannlaan 415</AdrLine>

<AdrLine>1180 Brussels</AdrLine>

<AdrLine>Belgium</AdrLine>

</PstlAdr>

</Dbtr>

### 3.1.7 MT\_To\_MXPartyNameAndStructuredAddress

**Name**

MT\_To\_MXPartyNameAndStructuredAddress

**Business description**

Every line in Subfield 2 (50F and 59F) starts with a number indicating the type of information provided. This function translates the structured information from the MT (with number “1/”, ”2/” and “3/”) to MX component equivalents by:

* checking the numbers to determine the MX target element
* 1/ Information is translated to MX Name after possible concatenation
* 2/ and 3/ Information are translated to MX Address Line as such with the number except 3/CountryCode/LEIC/LEIIdentifier is translated to MX LEI if the LEI has the correct pattern otherwise it is translated to MX AddressLine.
* Note that 3/CountryCode/LEIC/LEIIdentifier is allowed only for 59F. For 50F, the LEI is used with 6/ (see MX to MT translation). Using 3/CountryCode/LEIC/LEIIdentifier for 50F will generate issue in translation if 50F PartyIdentifier uses a code word defined in MT UHB (CUST, TXID, ..which are translated to Private ID while LEI is translated to OrganisationID. Private ID and Translation ID are exclusive)
* If the structure 3/CountryCode/LEIC/LEIIdentifier occurs multiple times (although it is not expected), the last occurrence will be the one translated.

**Format**

**MT\_To\_MXPartyNameAndStructuredAddress**(MTNameAndAdress, MXParty)

**Input**

MTNameAndAddress carries maximum 4 lines. Each of them starts with a number follows with “/” followed by information

**Output**

MXParty: the result of this function is immediately put into the correct MX element inside the target component typed *PartyIdentification135*.

No truncation to be foreseen

**Preconditions** None.

**Formal description**

/\*MTNameAndAddress is defined by max 4 lines of 35 characters.

MTNameAndAddress[1] indicates the first line\*/

/\* Local variables

i, index : integer

MXAddress []: table of string, max 3 \* 35 char

MXLEI : string \*/

index = 0

/\*Translation of the name present on line(s) starting with “1/”\*/

/\*Basic function *ExtractLines* extracts the name. The complete name could be present on several lines so the function is used with continuation string “1/”. Substring is taken to delete “1/” in the output string. If multiple lines are concatenated a space is added by the ExtractLines function. This is expected as information might carry different names split on several Lines in MT (eg, MT is the original message). If information comes from MX, the space might not be expected where translating back to MX but there is no way to know if information on several lines is on purpose or just the effect of long data requesting multiple lines\*/

MXParty.Name = **Substring(ExtractLines**(MTNameAndAddress, “1/”, “1/”), 3,140)

/\*Max 140 characters are copied to Name. No truncation\*/

/\*Translation of address details present on line(s) starting with “2/”. As per field 50F, 59F format specifications,”2/” must not be used without “3/” (Country, Town) but “3/” can be used without “2/” \*/

/\*Search for lines starting with “2/”. i > 1 but keep the loop starting at 1. \*/

**For** i = 1 to 4

**IF** **Substring**(MTNameAndAddress[i],1,2) = “2/” THEN

index = index + 1

/\* CRLF is removed before copying information to MX structure \*/

MXAddress[index]= MTNameAndAddress[i]

**ENDIF**

**END Loop**

/\* Search for lines starting with “3/”. If LEI is found and has the correct pattern, the line is translated to MX LEI \*/

**For** i = 1 to 4

**IF** **Substring**(MTNameAndAddress[i],1,2) = “3/” THEN

/\* Expected pattern “3/XY/LEIC/” \*/

**IF** **Substring**(MTNameAndAddress[i],1,10)has pattern “3/XY/LEIC/” where XY is 2 alphabetic characters (Is Country code but check on alphabetic is sufficient)

MXLEI = **Substring**(MTNameAndAddress[i],11)

**IF** MXLEI has pattern “[A-Z0-9]{18,18}[0-9]{2,2}” THEN

MXParty.Identification.OrganisationIdentification.LEI = MXLEI

~~Next i~~

**ELSE**

index = index + 1

/\* CRLF is removed before copying information to MX structure \*/

MXAddress[index]= MTNameAndAddress[i]

**ENDIF**

**ELSE**

Index = index + 1

/\* CRLF is removed before copying information to MX structure \*/

MXAddress[index]= MTNameAndAddress[i]

**ENDIF**

**ENDIF**

**Next i**

/\* Copy the MXaddress Information to MX AddressLine. Max 3 lines should be filled as “1/” is mandatory in MT and translated to MXName \*/

**IF** index > 0 THEN

**For** i = 1 to Index

MXParty.PostalAddress.AddressLine[i] = MXAddress[i]

**Next i**

**ENDIF**

### 3.1.8 MT\_To\_MXUltimateParty

**Name**

MT\_To\_MXUltimateParty

**Business description**

The function extracts information for Ultimate Debtor and Ultimate Creditor from field 70 in MT103 (RETN). This scenario can only occur if MT103 (RETN) results from a previous translation pacs.008 to MT103 or pacs.004 to MT103 RETN.

As described in MX\_To\_MTRemittanceInformation and in MX\_To\_MTRemittanceInformation2, the 2 code words /ULTB/ and /ULTD/ are **expected to be formatted as follows**:

The following code words will be used /ULTB/ to designate the ultimate creditor and /ULTD/ to designate the ultimate debtor.

/ULTB/ is followed by information about the UltimateCreditor BIC or (Name/Country [/TownName]), TownName is optional or (Name/OtherId) or Name alone or OtherId alone.

/ULTD/ is followed by information about the UltimateDebtor BIC or (Name/Country/TownName), TownName is mandatory or (Name/OtherId) or Name alone or OtherId alone.

It will not be possible to distinguish between Name alone and OtherID alone. So they will be both translated to Name.

IF BIC is present, it is assumed, per MX to MT translation that there is no other data related to the party codeword.

**Parsing Assumption** is that there is no “/” inside the Name or inside OtherId otherwise the parsing will produce incorrect results.

**Format**

**MT\_To\_MXUltimateParty**(MTField70, MXLocation)

**Input**

MTField70 : Field70 in MT103

**Output**

MXLocation is generic and must be adapted depending on the target message type:

-In pacs.008: MXLocation = MX CreditTransferTransactionInformation typed *CreditTransferTransaction39*

-In pacs.004 : MXLocation = MXReturnChain: element typed *TransactionParties7*

**Preconditions**

**The formats as described in the business description are required and should be a consequence of MX to MT previous translation.**

(These preconditions cannot be implemented).

**Formal description**

/\* Local variables

DebtorString, CreditorString, PartyString : string

DebtorPattern, CreditorPattern, ToDeletePattern : string

MXName, MXCountry, MXTownName, MXOtherId, MXBIC, PathUCreditor, PathUDebtor : string

PatternTable[] : table of string

i : integer

\*/

DebtorPattern = “/ULTD/”

CreditorPattern = “/ULTB/”

ToDeletePattern = “//”

IF MXLocation = “MXReturnChain” THEN

/\* Target is pacs.004 \*/

PathUCreditor = MXLocation.UltimateCreditor.Party

PathUDebtor = MXLocation.UltimateDebtor.Party

ELSE

/\* Target is pacs.008 \*/

PathUCreditor = MXLocation.UltimateCreditor

PathUDebtor = MXLocation.UltimateDebtor

ENDIF

/\* Search for the codewords /ULTB/ and /ULTD/ and extract information following the codewords.

Possible other codewords present in field 70 : /PURP/, /ROC/, /URI/, /RELID/, /SRI/. Note that for the target message pacs.004, only /ULTB/ and /ULTD/ are expected due to previous translation MX to MT \*/

PatternTable[1] = “/ULTB/”

PatternTable[2] = “/ULTD/”

PatternTable[3] = “/PURP/”

PatternTable[4] = “/ROC/”

PatternTable[5] = “/URI/”

PatternTable[6] = “/RELID/”

PatternTable[7] = “/SRI/”

/\* Other possible codewords in Field 70 are added to improve data extraction \*/

PatternTable[8] = “/INV/”

PatternTable[9] = “/IPI/”

PatternTable[10] = “/RFB/”

PatternTable[11] = “/ROC/”

PatternTable[12] = “/TSU/”

CreditorString= **ExtractBetweenPattern**(MTField70,CreditorPattern,

{PatternTable[1], PatternTable[2], PatternTable[3], PatternTable[4], PatternTable[5], PatternTable[6], PatternTable[7], PatternTable[8], PatternTable[9], PatternTable[10], PatternTable[11], PatternTable[12]})

/\* The additional “//” are removed if they occur at the end of the extracted information. They would occur if the second pattern is found due to the construction of the translation MX to MT separating code words with “//”

For example, possible structures : /ULTB/CreditorName///URI/RemittanceInfo OR /ULTB/CreditorName In the first example, the additional “//” must be removed, not in the second case. \*/

Call **SubfunctionTrimRight**(CreditorString, CreditorPattern, MTField70,ToDeletePattern, PatternTable[]; CreditorString)

DebtorString= **ExtractBetweenPattern**(MTField70,DebtorPattern,

{PatternTable[1], PatternTable[2], PatternTable[3], PatternTable[4], PatternTable[5], PatternTable[6], PatternTable[7],PatternTable[8], PatternTable[9], PatternTable[10], PatternTable[11], PatternTable[12]})

Call **SubfunctionTrimRight**(DebtorString, DebtorPattern, MTField70,ToDeletePattern, PatternTable[]; DebtorString)

/\* Parse the CreditorString and copy to MX elements \*/

**IF** **Length**(CreditorString) > 0 THEN

PartyString = CreditorString

Call **SubfunctionExtractInformation**(PartyString;MXName, MXCountry, MXTownName, MXOtherId, MXBIC)

/\* Subfonction described below \*/

**IF Length**(MXBIC)> 0 THEN

PathUCreditor.Identification.OrganisationIdentification.AnyBIC = MXBIC

**ELSE**

**{**

**IF Length**(MXName) > 0 THEN

/\* MXName should always get a value either the Name or Other Id \*/

**IF** Length(MXName)> 140 THEN

MXName = **Concatenate**(**Substring**(MXName, 1,139),”+)

**ENDIF**

PathUCreditor.Name = MXName

**IF Length**(MXCountry)> 0 AND **IsValidCountryCode**(MXCountry) > 0 THEN

/\* Must be a country code in the ISO list \*/

PathUCreditor.PostalAddress.Country = MXCountry

/\* TownName is copied only if valid country code \*/

**IF Length**(MXTownName)>0 THEN

**IF Length**(MXTownName)> 35 THEN

MXTownName = **Concatenate**(Substring(MXTownName,1,34), “+”)

**ENDIF**

PathUCreditor.PostalAddress.TownName = MXTownName

**ENDIF**

**ELSE** /\* No MXCountry \*/

**IF Length**(MXOther) > 0 THEN

/\* Copy to Private ID \*/

**IF Length**(MXOther) > 35 THEN

MXOther = **Concatenate**(Substring(MXOther,1,34), “+”)

**ENDIF**

PathUCreditor.Identification.PrivateIdentification.Other.Identification = MXOther

**ENDIF**

**ENDIF** /\* Length MX Country \*/

**ENDIF** /\* Length MX Name \*/

**}**

**ENDIF** /\* Length MX BIC \*/

**ENDIF** /\* Length CreditorString \*/

/\* Parse the DebtorString and copy to MX elements. Simiilar logic as above but both country and TownName must be present otherwise they will not copied to MX \*/

**IF Length**(DebtorString) > 0 THEN

PartyString = DebtorString

**Call** **SubfunctionExtractInformation**(PartyString;MXName, MXCountry, MXTownName, MXOtherId, MXBIC)

**IF Length**(MXBIC)> 0 THEN

PathUDebtor.Identification.OrganisationIdentification.AnyBIC = MXBIC

**ELSE**

**{**

**IF Length**(MXName) > 0 THEN

/\* MXNAme should always get a value either the Name or Other Id \*/

**IF** Length(MXName)> 140 THEN

MXName = **Concatenate**(**Substring**(MXName, 1,139),”+)

**ENDIF**

PathUDebtor.Name = MXName

**IF Length**(MXCountry)> 0 AND **IsValidCountryCode**(MXCountry) > 0 THEN

/\* Must be a country code in the ISO list \*/

/\* TownName is copied only if valid country code \*/

**IF Length**(MXTownName)>0 THEN

**IF Length**(MXTownName)> 35 THEN

MXTownName = **Concatenate**(Substring(MXTownName,1,34), “+”)

**ENDIF**

**ELSE** /\* Valid country but no TownName \*/

**MXTownName = “NOTPROVIDED”**

**ENDIF** /\* Length MXTownName \*/

PathUDebtor.PostalAddress.Country = MXCountry

PathUDebtor.PostalAddress.TownName = MXTownName

**ELSE** /\* No MXCountry \*/

**IF Length**(MXOther) > 0 THEN

/\* Copy to Private ID \*/

**IF Length**(MXOther) > 35 THEN

MXOther = **Concatenate**(Substring(MXOther,1,34), “+”)

**ENDIF**

PathUDebtor.Identification.PrivateIdentification.Other.Identification = MXOther

**ENDIF**

**ENDIF** /\* Length MX Country \*/

**ENDIF** /\* Length MX Name \*/

**}**

**ENDIF** /\* Length MX BIC \*/

**ENDIF** /\* Length DebtorString \*/

**SubfunctionTrimRight**(Value,FromPattern, SearchInString, ToDeletePattern, PatternTable[];Value)

/\* Local variables

i : integer

ToTrimRight : Boolean \*/

ToTrimRight = “false”

/\* Check if the data extracted is followed by another codeword or not. If yes, “//” has to be deleted \*/

SearchInString = **ExtractFromPattern**(SearchInString, FromPattern)

/\* Select the string after FromPattern \*/

**IF Length**(SearchInString)> 0 THEN

**For** i = 1 To **NumberOfOccurrences**(PatternTable[])

**IF IsPresentPattern**(SearchInString, PatternTable[i])THEN

ToTrimRight = “true”

EXIT loop

**ELSE**

Next i

**ENDIF**

**END Loop**

**IF** ToTrimRight THEN

/\* Remove the “//” added to separate MX codewords in MX to MT translation \*/

**IF Substring**(Value,Length(Value)-1,2) = ToDeletePattern

/\* Remove the last 2 characters \*/

Value = **Substring**(Value,1,**Length**(Value)-2)

**ENDIF**

**ENDIF**

**ENDIF**

**End SubfunctionTrimRight**

**SubfunctionExtractInformation**(PartyString; MXName, MXCountry, MXTownName, MXOtherId, MXBIC)

/\* The subfunction parses the PartyString and retrieve MXBIC or MXName, MXCountry, MXTownName, MXOtherId if present. **Assumption is that “/” is not used as part of the data but only as a separator between the data otherwise parsing will not work properly**

**IF BIC is found, the search stops** \*/

/\* Note : IF there is some risk “/” is used in business data, another separator like “(“ could be used in MX to MT and MT to MX functions related to UltimateParties \*/

/\* Local variables

PatternNameCountryTownName, PatternNameOtherOrCountry : string

SeparatorChar : char \*/

PatternNameCountryTownName = “\*/[A-Z]{2}/\*”

/\* where “\*” means any characters other than “/” \*/

PatternNameOtherOrCountry = “\*/\*”

SeparatorChar = “/”

MXName = “”

MXCountry = “”

MXTownName = “”

MXOther = “”

MXBIC = “”

**Case 1** /\* Search for valid BIC, meaning the structure of the BIC is correct and the BIC (ie AnyBIC type) exists \*/

**IF** **IsValidBIC**(PartyString) THEN

MXBIC = PartyString

Return

**ELSE**

Call Case 2

**ENDIF**

**Case 2**

**IF IsPresentPattern**(PartyString, PatternNameCountryTownName) THEN

/\* Search for pattern “Name/Country/TownName” \*/

/\* TillPattern is not extracted in the return string \*/

/\* Name - Extract information until “/CountryCode/” pattern \*/

MXName = **ExtractTillPattern**(PartyString,**Concatenate**(SeparatorChar),[A-Z]{2},SeparatorChar))

MXCountry = **ExtractTillPattern**(PartyString,Length(MXName)+2, SeparatorChar)

**IF NOT IsValidCountryCode**(MXCountry)THEN

MXCountry = “”

**T12006**

**ENDIF**

MXTownName = **ExtractFromPattern**(PartyString,Length(MXName) + Length(MXCountry)+1 , SeparatorChar)

**ELSEIF IsPresentPattern**(PartyString, PatternNameOtherOrCountry) THEN

/\* Search for a pattern “Name/Other” or Name/Country \*/

MXName = **ExtractTillPattern**(PartyString,SeparatorChar)

MXOther = **ExtractFromPattern**(PartyString, Length(MXName), SeparatorChar)

**IF IsValidCountryCode**(MXOTHER) THEN

/\* For UltimateCreditor, TownName is optional, so with Name either the country code can be provided or Other ID \*/

MXCountry = MXOther

MXOther = “”

**ENDIF**

**ELSE**

MXName = PartyString

**ENDIF**

## 3.2 Financial Institution Translation Rule Descriptions

The translation rule descriptions provided in this section are for translation rules that relate to financial institution information.

### 3.2.1 MT\_To\_MXBICFI

**Name**

MT\_To\_MXBICFI

**Business description**

The function translates an MT BICFI to an MX BICFI.

**Format**

**MT\_To\_MXBICFI**(MTBICFI, ; MXAgent)

**Input**

MTBICFI: BIC identifying a financial institution in an MT message.

**Output**

MXAgent: the result of this function -a BIC- is immediately put into the correct MX element inside the target component typed *BranchAndFinancialInstitutionIdentification6*.

**Preconditions** None.

**Formal description**

MXAgent.FinancialInstitutionIdentification.BICFI = MTBIC

**Example 1 MT Source:**

:52A: *//BL12345678*

SOGEDEFF

*(Italic not translated by this function)*

**MX Translation:**

<DbtrAgt>

<FinInstnId>

<BICFI>SOGEDEFF</BICFI>

<*ClrSysMmbId>*

*<ClrSysId>*

*<Cd>DEBLZ</Cd>*

*</ClrSysId>*

*<MmbId>12345678</MmbId>*

*</ClrSysMmbId>*

</FinInstnId>

</DbtrAgt>

**Example 2 MT Source:** :57A:*/123456789*  IRVTUS3N

*(Italic not translated by this function)*

**MX Translation:**

<CdtrAgt>

<FinInstnId>

<BICFI>IRVTUS3N</BICFI>

</FinInstnId>

</CdtrAgt>

**Example 3 MT Source:**

:57A:IRVTUS33

**MX Translation:**

<CdtrAgt>

<FinInstnId>

<BICFI>IRVTUS33</BICFI>

</FinInstnId>

</CdtrAgt>

### 3.2.2 MT\_To\_MXClearingIdentifier

**Name**

MT\_To\_MXClearingIdentifier

**Business description**

The function translates an MT clearing identifier to an MX clearing system member identification by:

* deleting the double slash ‘//’ preceding the MT clearing identifier. In case the MT clearing identifier is preceded by a clearing channel indicator “//RT” (pay through a real time gross settlement system), this prefix is also deleted (translation of the clearing channel indicator is handled elsewhere).
* isolating the first two characters in the MT clearing identifier that identify the MT clearing system from the MT clearing code that identify the financial institution in the system.
* checking whether the MT clearing system has an equivalent in the ISO 20022 externalised clearing system list (see the Annex to this function, below the function description). If an equivalent code is available, the clearing system is translated to its MX equivalent in the target element

”ClearingSystemMemberIdentification.ClearingSystemIdentification.Code”.

All the MT Clearing codes listed in the MT generic list (UHB) are included in the ISO Clearing code list. IF the MTClearingSystem has no equivalent in the ISO 20022 externalised clearing system list or if the Clearing system member identification is absent, it is not translated in this function

* The Identification of the Financial Institution in the system, will be written “as is” in the ClearingSystemMemberIdentification.MemberIdentification element.

IF the payment is originated in MX, if the MX ClearingSystemIdentification (5 char) has no MT equivalent (2 char), the MX ClearingSystemIdentification (5 char) has been copied as such in the MTPartyIdentifier after concatenation with the MemberID. Therefore it can be translated back to MX ClearingSystemIdentification. That case is also covered in the function.

**Format**

**MT\_To\_MXClearingIdentifier**(MTPartyIdentifier, ; MXClearingIdentifier)

**Input**

MTPartyIdentifier: clearing system member identification of a financial institution in the MT message.

**Output**

MXClearingIdentifier: the result of this function -the MX clearing system member identification- is immediately put into the correct MX elements inside the target component typed *BranchAndFinancialInstitutionIdentification6*.

**Preconditions** None.

**Formal description**

/\* Check first ifMX ClearingSystemIdentification (5 char) has been copied as such in the MTPartyIdentifier after concatenation with the MemberID \*/

/\*Find clearing system and clearing code by taking out the double slash “//” or clearing channel indicator and double slash. MTClearingSystem, MTClearingCode and MXClearingSystem are local variables\*/

**IF** **Substring**(MTPartyIdentifier, 1, 6) = “//RT//” THEN

MTClearingSystem = **Substring**(MTPartyIdentifier, 7, 5)

MTClearingCode = **Substring**(MTPartyIdentifier, 12)

**ELSE**

MTClearingSystem = **Substring**(MTPartyIdentifier, 3, 5)

MTClearingCode = **Substring**(MTPartyIdentifier, 8)

**ENDIF**

**IF Length**(**ExtractPattern**(MTClearingSystem, [A-Z]{5})) > 0 AND **Length**(MTClearingCode) > 0 AND **IF** **WithinList**(MTClearingSystem, ExternalClearingSystemIdentification1Code)

/\* ExternalClearingSystemIdentification1Code is published by ISO2022 – Sheet 5 in the ISO excel document \*/

THEN

FinancialInstitutionIdentification.ClearingSystemMembe rIdentification.ClearingSystemIdentification.Code =

MTClearingSystem

FinancialInstitutionIdentification.ClearingSystemMemberIden tification.MemberIdentification = MTClearingCode

EXIT Function

/\* Else continue the search \*/

**ENDIF**

/\* Search a FIN structure for the ClearingSystemIdentification with 2 char \*/

**IF** **Substring**(MTPartyIdentifier, 1, 6) = “//RT//” THEN

MTClearingSystem = **Substring**(MTPartyIdentifier, 7, 2)

MTClearingCode = **Substring**(MTPartyIdentifier, 9)

**ELSE**

MTClearingSystem = **Substring**(MTPartyIdentifier, 3, 2)

MTClearingCode = **Substring**(MTPartyIdentifier, 5)

**ENDIF**

/\*Check whether the MTClearingSystem belongs to the

MTClearingSystemList with an equivalent in the ISO MXClearingSystemList. IF MTClearingSystem is empty, it is treated as not found in the list\*/

**IF** **WithinList**(MTClearingSystem, MTClearingSystemList)AND NOT **IsEmpty**(MTClearingCode)

/\* MTClearingSystemList is described below in annex section \*/

/\*Target will be the

ClearingSystemMemberIdentification.ClearingSystemIdentifica tion.Code element\*/

/\*Exception handling when MTClearingSystem equals “SW”: 2 target ISO MXClearingSystem codes correspond to the same code “SW”. Depending on the length of the MTClearingCode it will be one or the other but this check is not handled by the *EquivalentCode* function where a one on one relation is needed\*/

**IF** MTClearingSystem = “SW” AND **Length**(MTClearingCode) > 5

MXClearingSystem = “CHSIC” THEN

**ELSEIF** MTClearingSystem = “SW” THEN

MXClearingSystem = “CHBCC”

**ELSE**

/\*Replace MTClearingSystem with MX equivalent in the

ISO MXClearingSystemList\*/

MXClearingSystem = **EquivalentCode**(MTClearingSystem,

MTClearingSystemList, MXClearingSystemList)

**ENDIF**

/\* Write output string in their MX target elements \*/

FinancialInstitutionIdentification.ClearingSystemMembe rIdentification.ClearingSystemIdentification.Code =

MXClearingSystem

FinancialInstitutionIdentification.ClearingSystemMemberIden tification.MemberIdentification = MTClearingCode

/\*IF the MTClearingSystem is not in the list or the MTClearingCode is absent, it is not translated. \*/

**ELSE**

/\*No translation \*/

**ENDIF** /\* ENDIF WithinList \*/

**For MT and ISO 20022 “Externalised” MX Clearing System Lists, please refer to section 5 Annex.**

**Example 1 MT Source:**

:52A://BL12345678

*SOGEDEFF*

*(Italic not translated by this function)*

**MX Translation:**

<DbtrAgt>

<FinInstnId>

*<BICFI>SOGEDEFF</BICFI>*

<ClrSysMmbId>

<ClrSysId>

<Cd>DEBLZ</Cd>

<ClrSysId>

<MmbId>12345678</MmbId>

</ClrSysMmbId>

</FinInstnId>

</DbtrAgt>

**Example 2 MT Source:**

:57C://FW123456789

**MX Translation:**

<CdtrAgt>

<FinInstnId>

<ClrSysMmbId>

<ClrSysId>

<Cd>USABA</Cd>

</ClrSysId>

<MmbId>123456789</MmbId>

</ClrSysMmbId>

</FinInstnId>

</CdtrAgt>

**Example 3 MT Source:**

:57D*://BE1234572*

*BANK ABC*

*BE/UTOPIA*

*(Italic not translated by this function)*

*/\* In case of option A and D, and if there is no ISO equivalent clearingSystemIdentifier, this information is not translated /\**

**MX Translation:**

<CdtrAgt>

<FinInstnId>

*<Nm>BANK ABC</Nm>*

*<PstlAdr>*

*<AdrLine>BE/UTOPIA</AdrLine> </PstlAdr>*

</FinInstnId>

</CdtrAgt>

### 3.2.3 IsMTClearingSystemCodeInList

**Name**

IsMTClearingSystemCodeInList

**Business description**

The function looks in the MT to MX clearing system code generic table if the MTClearingSystemCode is present and the MTClearingSystemMemberIdentification is not empty: value “true” is returned else “false”. This function will be called when an FI is identified trough a format option C and D.

IF the payment is originated in MX, if the MX ClearingSystemIdentification (5 char) has no MT equivalent (2 char), the MX ClearingSystemIdentification (5 char) has been copied as such in the MTPartyIdentifier after concatenation with the MemberID. Therefore it can be translated back to MX ClearingSystemIdentification. That case is also checked in the function and will return “true”.

**Format**

**IsMTClearingSystemCodeinList**(MTPartyIdentifier;)

**Input**

MTPartyIdentifier: clearing system member identification of a financial institution in the MT message.

**Output**

Boolean is returned

**Preconditions** None.

**Formal description**

/\* Check first ifMX ClearingSystemIdentification (5 char) has been copied as such in the MTPartyIdentifier after concatenation with the MemberID \*//\*Find clearing system and clearing code by taking out the double slash “//” or clearing channel indicator and double slash. MTClearingSystem, MTClearingCode and MXClearingSystem are local variables\*/

**IF** **Substring**(MTPartyIdentifier, 1, 6) = “//RT//” THEN

MTClearingSystem = **Substring**(MTPartyIdentifier, 7, 5)

MTClearingCode = **Substring**(MTPartyIdentifier, 12)

**ELSE**

MTClearingSystem = **Substring**(MTPartyIdentifier, 3, 5)

MTClearingCode = **Substring**(MTPartyIdentifier, 8)

**ENDIF**

**IF Length**(**ExtractPattern**(MTClearingSystem, [A-Z]{5})) > 0 AND **Length**(MTClearingCode) > 0 AND **IF** **WithinList**(MTClearingSystem, ExternalClearingSystemIdentification1Code)

/\* ExternalClearingSystemIdentification1Code is published by ISO2022 – Sheet 5 in the ISO excel document \*/

THEN

Return “True”

/\* Else continue the search \*/

**ENDIF**

/\* Search a FIN structure for the ClearingSystemIdentification with 2 char \*/

**IF** **Substring**(MTPartyIdentifier, 1, 6) = “//RT//” THEN

MTClearingSystem = **Substring**(MTPartyIdentifier, 7, 2)

MTClearingCode = **Substring**(MTPartyIdentifier, 9)

**ELSE**

MTClearingSystem = **Substring**(MTPartyIdentifier, 3, 2)

MTClearingCode = **Substring**(MTPartyIdentifier, 5)

**ENDIF**

/\*Check whether the MTClearingSystem belongs to the

MTClearingSystemList with an equivalent in the ISO MXClearingSystemList. IF MTClearingSystem is emty, it is treated as not found in the list\*/

**IF** **WithinList**(MTClearingSystem, MTClearingSystemList)AND NOT **IsEmpty**(MTClearingCode) THEN

Return “true”

**Else**

Return “false”

**ENDIF**

### 3.2.4 MT\_To\_MXClearingSystemToNameAndAddressLine

**Name**

MT\_To\_MXClearingSystemToNameAndAddressLine

**Business description**

This function is used when the MT Clearing System Code Identifier has no equivalent ISO code and there is no other identification for the FI (ie., no BIC and No Name and Address). In MX, the ClearingSystemIdentification only allows ISO code as proprietary identification is removed. Although it is no a clean solution, the MT Clearing System Code Identifier is copied in Name AND in the AddressLine[1] in order to easily identify it is a special case. The MT “//” identifying a ClearingSystemMemberId is kept in MX also to ease the identification of the data.

**Format**

**MT\_To\_MXClearingSystemToNameAndAddressLine**(MTPartyIdentifier; MXPartyIdentifier)

**Input**

MTPartyIdentifier: clearing system member identification of a financial institution in the MT message.

**Output**

MXPartyIdentifier defined as *BranchAndFinancialInstitutionIdentification6* data type

**Preconditions**

Function call when the FI format is option C or B

**Formal description**

If the Clearing system code information is not empty and is not equal to “RT”, it is copied to Party Name and AddressLine. Otherwise PartyName and AddressLine are filled with value “NOTPROVIDED”.

/\* Local variables:

MTClearingSystemMemberIdentification : string

\*/

/\*Find clearing system and clearing code by taking out the double slash “//” or clearing channel indicator and double slash. MTClearingSystem, MTClearingCode and MXClearingSystem are local variables\*/

**IF** **Substring**(MTPartyIdentifier, 1, 6) = “//RT//” THEN

MTClearingSystemMemberIdentification = **Substring**(MTPartyIdentifier, 7)

**ELSE**

MTClearingSystemMemberIdentification = **Substring**(MTPartyIdentifier, 3)

**ENDIF**

**IF** Not**IsEmpty**(MTClearingSystemMemberIdentification)AND MTClearingSystemMemberIdentification NOT Equal to “RT” THEN

/\* Restore “//” in MXClearingSystemMemberID \*/

MXParty.FinancialInstitutionIdentification.Name = **Concatenate**(“//”, MTClearingSystemMemberIdentification)

MXParty.FinancialInstitutionIdentification.PostalAddress.AddressLine[1] = MXParty.FinancialInstitutionIdentification.Name

T11002 /\* Error code described in the error code list \*/

**ELSE**

MXParty.FinancialInstitutionIdentification.Name = “NOTPROVIDED”

MXParty.FinancialInstitutionIdentification.PostalAddress.AddressLine = “NOTPROVIDED”

T11003 /\* Error code described in the error code list \*/

**ENDIF**

### 3.2.5 MT\_To\_MXFinancialInstitutionNameAndUnstructuredAddress

**Name**

MT\_To\_MXFinancialInstitutionNameAndUnstructuredAddress

**Business description**

The function translates an MT financial institution name and address (4 \*35) to an MX name and MX address part of the *BranchAndFinancialInstitutionIdentification6* data type. The first line from MT financial name and address is copied to MX Name, the next 3 lines are copied to MX AddressLine (3 \* 35 char).

Default value for AddressLine is provided. Except National Payments (Debtor Agent & Creditor Agent within same country), Name and PostalAddress are mandatory if BIC is absent. So value « NOTPROVIDED » will be filled in any cases if MT PostalAddress is absent.

**Format**

**MT\_To\_MXFinancialInstitutionNameAndUnstructuredAddress**(MTNameAndAddress ; MXAgent)

**Input**

MTNameAndAddress: name and address of a financial institution in an MT message.

**Output**

MXAgent: the result of this function is immediately put into the correct MX element inside the target component typed *BranchAndFinancialInstitutionIdentification6*.

**Preconditions** None.

**Formal description**

/\*MTNameAndAddress is defined by 4 lines of 35 characters.

MTNameAndAddress[1] indicates the first line not including the Carriage Return Line Feed “*CRLF*” needed as separator between

consecutive lines in an MT field with multiple line format\*/

/\* Local variable

Path : string \*/

Path = MXAgent. FinancialInstitutionIdentification

Path.Name = MTNameAndAddress[1]

/\*Translation of the address details. AddressLine in the target

MX PostalAddress componenent is repetitive. PostalAddress.AddressLine[1] indicates the first occurrence etc.\*/

**IF** **IsPresent**(MTNameAndAddress[4])THEN

Path.PostalAddress.AddressLine[1] = MTNameAndAddress[2]

Path.PostalAddress.AddressLine[2] = MTNameAndAddress[3]

Path.PostalAddress.AddressLine[3] = MTNameAndAddress[4]

**ELSEIF** **IsPresent**(MTNameAndAddress[3])THEN

Path.PostalAddress.AddressLine[1] = MTNameAndAddress[2]

Path.PostalAddress.AddressLine[2] = MTNameAndAddress[3]

**ELSEIF** **IsPresent**(MTNameAndAddress[2])THEN

Path.PostalAddress.AddressLine[1] = MTNameAndAddress[2]

**ENDIF**

/\* Default values for AddressLine. With some exceptions like National Payments (Debtor Agent & Creditor Agent within same country), Name and PostalAddress are mandatory if BIC is absent. So value « NOTPROVIDED » will be filled in in any cases if MT PostalAddress is absent. \*/

IF (Path.PostalAddress.AddressLine[1] AND Path.PostalAddress.AddressLine[2]AND Path.PostalAddress.AddressLine[3]) **IsEMPTY** THEN

Path.PostalAddress.AddressLine[1] = “NOTPROVIDED”

ENDIF

**Example 1 MT Source:**

:52D:*//FW123456789*

Bank ABC

Wall Street 1

US/New York

*(Italic not translated by this function)*

**MX Translation:**

<DbtrAgt>

<FinInstnId>

*<ClrSysMmbId>*

*<ClrSysId>*

*<Cd>USABA</Cd>*

*<ClrSysId>*

*<MmbId>123456789</MmbId>*

*</ClrSysMmbId>*

<Nm>Bank ABC</Nm>

<PstlAdr>

<AdrLine>Wall Street 1</AdrLine>

<AdrLine>US/New York</AdrLine>

</PstlAdr>

</FinInstnId>

</DbtrAgt>

**Example 2 MT Source:**

:57D:*/123456789*

Bank ABC

US

*(Italic not illustrated by this function)*

**MX Translation:**

<CdtrAgt>

<FinInstnId>

<Nm>Bank ABC</Nm>

<PstlAdr>

<AdrLine>US</AdrLine>

</PstlAdr>

</FinInstnId>

</CdtrAgt>

**Example 3 MT Source:**

:57D:Bank ABC

US

**MX Translation:**

<CdtrAgt>

<FinInstnId>

<Nm>Bank ABC</Nm>

<PstlAdr>

<AdrLine>US</AdrLine>

</PstlAdr>

</FinInstnId>

</CdtrAgt>

**Example 4 MT Source:**

:52D:Bank ABC

Wall Street 1

New York

**MX Translation:**

<DbtrAgt>

<FinInstnId>

<Nm>Bank ABC</Nm>

<PstlAdr>

<AdrLine>Wall Street 1</AdrLine>

<AdrLine>New York</AdrLine>

</PstlAdr>

</FinInstnId>

</DbtrAgt>

### 3.2.6 MT72INS\_To\_MXAgent

**Name**

MT72INS\_To\_MXAgent

**Business description**

The function extracts the information following a code /INS/ identifying an instructing institution in a field 72, checks whether the string contains a BIC and translates accordingly to the appropriate element of an MX previous instructing agent element (BIC or Name). As up to 3 MX previous instructing agents are allowed, information must be filled in in the same order as the /INS/ occurrences occur in MT.

If more than 3 occurrences are present in field 72, they will be ignored in translation.

If more than 3 occurrences of /INS/ are present, an error/warning message is displayed.

If the Name is longer than 140 char, a sign “+” is added at the end of the string to indicate truncation.

IF Name is extracted, PostalAddress.AddressLine gets dummy value “NOTPROVIDED”.

Assumption : MX Agent only allows FI BIC. So the BIC following /INS/ must be a FI BIC. Using a non FI BIC after /INS/ will generate an invalid MX message.

**Format**

**MT72INS\_To\_MXAgent**(MT72 ; MXAgent1Id, MXAgent2Id, MXAgent3Id)

**Input**

MT72: content of an MT field 72 with format 6\*35x (6 lines of 35 characters).

**Output**

MXAgent1Id, MXAgent2Id, MXAgent3Id: the result of this function is immediately put into the correct MX element inside the target component typed *BranchAndFinancialInstitutionIdentification6*.

**Preconditions** None.

**Formal description**

/\*Field 72 is defined by 6 lines of 35 characters with “//” as line continuation pattern and line separator “*CRLF*” between consecutive lines.

Basic function *ExtractLines* extracts the information with code /INS/. The function removes the MT line separator "CRLF" and replaces the continuation pattern "//" by a space “ ”.

MT72Extract1,2,3 and RemainingLines are local variables. Substring is taken to delete “/INS/” in the output string.\*/

/\* Local Variables

MT72Extract1,2,3,4 and RemainingLines: string

INSCode: Boolean \*/

INSCode = “false”

/\*Extract first occurrence of /INS/ \*/

MT72Extract1 = **Substring(ExtractLines**(MT72, “/INS/”, “//”), 6)

**IF** **IsBIC**(MT72Extract1)

MXAgent1.FinancialInstitutionIdentification.BIC = MT72Extract1

**ELSEIF** **Length**(MT72Extract1)> 0 THEN

/\*Restriction of the Name to maximum 140 characters\*/

**IF** **Length**(MT72Extract1) > 140 THEN

MT72Extract1 = **Concatenate**(**Substring**(MT72Extract1, 1, 139),”+”)

**END IF**

MXAgent1.FinancialInstitutionIdentification.Name =

**Substring**(MT72Extract1, 1, 140)

MXAgent1.FinancialInstitutionIdentification.PostalAddress.AddressLine = “NOTPROVIDED”

**ENDIF** /\* IF LENGTH (MT72Extract1)> 0\*/

/\* search for a second occurrence of /INS/ \*/

RemainingLines = **DeleteExtractLines**(MT72, “/INS/”, “//”)

/\*DeleteExtractLines remove from the source the information extracted with the function ExtractLines \*/

**IF** **Length**(RemainingLines) > 0 Then

MT72Extract2 = **Substring(ExtractLines**(RemainingLines, “/INS/”, “//”), 6)

**IF** **IsBIC**(MT72Extract2)

MXAgent2.FinancialInstitutionIdentification.BIC = MT72Extract2

**ELSEIF** **Length**(MT72Extract2)> 0 THEN

/\*Restriction of the Name to maximum 140 characters\*/

\*/

**IF** **Length**(MT72Extract2) > 140 THEN

MT72Extract2 = **Concatenate**(**Substring**(MT72Extract2, 1, 139),”+”)

**ENDIF**

MXAgent2.FinancialInstitutionIdentification.Name =

**Substring**(MT72Extract2, 1, 140)

MXAgent2.FinancialInstitutionIdentification.PostalAddress.AddressLine = “NOTPROVIDED”

**ENDIF** /\* IF Length(MT72Extract2)> 0 \*/

**ENDIF** /\* Length (RemainingLines) > 0 \*/

RemainingLines = **DeleteExtractLines**(RemainingLines, “/INS/”, “//”)

/\* search for a third occurrence of /INS/ \*/

**IF** **Length**(RemainingLines) > 0 Then

MT72Extract3 = **Substring(ExtractLines**(RemainingLines, “/INS/”, “//”), 6)

**IF Length**(MT72Extract3)> 0 THEN

INSCode = “True”

**ENDIF**

**IF** **IsBIC**(MT72Extract3)

MXAgent3.FinancialInstitutionIdentification.BIC = MT72Extract3

**ELSEIF** **Length**(MT72Extract3)>0 THEN

/\*Restriction of the Name to maximum 140 characters\*/

**IF** **Length**(MT72Extract3) > 140 THEN

MT72Extract3 = **Concatenate**(**Substring**(MT72Extract3,1,139),”+”)

**ENDIF**

MXAgent3.FinancialInstitutionIdentification.Name =

**Substring**(MT72Extract3, 1, 140)

MXAgent3.FinancialInstitutionIdentification.PostalAddress.AddressLine = “NOTPROVIDED”

**ENDIF** /\* IF Length(MT72Extract3)> 0\*/

**ENDIF** /\* IF Length(RemainingLines) > 0 \*/

/\* Check if there is a 4th INS code. In which case an warning message is reported. This is to cater for the case where from pacs.009 DebtorAgent is translated to /INS/ and therefore 4 occurrences of /INS/ are possible or original payment in MT has more than 3 occurrences in /INS/. From MT, it is not possible to identify if the first occurrence of /INS/ has to be translated to DebtorAgent in pacs.009 as a side effect from MX to MT translation. \*/

**IF** INSCode THEN

/\* delete the 3rd INS occurrence information \*/

RemainingLines = **DeleteExtractLines**(RemainingLines, “/INS/”, “//”)

**IF** **Length**(RemainingLines) > 0 Then

MT72Extract4 = **Substring(ExtractLines**(RemainingLines, “/INS/”, “//”), 6)

**IF Length**(MT72Extract4) > 0 THEN

Error/Warning Message

/\* as the 4th occurrence and the next ones are never translated, “Ignored” should be reported. But for technical reasons, it is too difficult to implement such code. So it will be replaced by a code in the category TRUNC Missing \*/

**ENDIF**

**ENDIF**

**ENDIF** /\* IF INSCode \*/

**Example 1 MT Source:** :72:/INS/GKCCBEBB

**MX Translation:**

<PrvsInstgAgt1>

<FinInstnId>

<BICFI>GKCCBEBB</BICFI>

</FinInstnId>

</PrvsInstgAgt1>

**Example 2 MT Source:**

:72:/INS/Bank ABC

**MX Translation:**

<PrvsInstgAgt1>

<FinInstnId>

<Nm>Bank ABC</Nm>

<PstlAdr>

<AdrLine>NOTPROVIDED</AdrLine>

</PstlAdr>

</FinInstnId>

</PrvsInstgAgt1>

### 3.2.7 MT\_To\_MXFinancialInstitutionAccount

**Name**

MT\_To\_MXFinancialInstitutionAccount

**Business description**

The function translates an MT financial institution account to an MX account by deleting the slash “/”, or '//CH' (in case of a CHIPS Universal Identifier), the debit indicator and slash “/D/” or the credit indicator and slash “/C/” and writing the remainder in the appropriate MX account element in Identification/Other/Identification or Identification/IBAN.

The function handles also the case where only /C or /D is provided with no account identification. In that case MX Account is not populated. The METAFCTxxx in excel handles the translation of /C or /D to SettlementMethod if present in 53a. Otherwise /C or /D is ignored.

**Format**

**MT\_To\_MXFinancialInstitutionAccount**(MTPartyIdentifier ; MXAccount)

**Input**

MTPartyIdentifier: financial institution account in the MT format [/1!a]/34x ([ ] indicate optionality).

**Output**

MXAccount: the result of this function -an MX account- is immediately put into the correct MX element inside the target component typed *CashAccount38*.

**Preconditions** None.

**Formal description**

/\* Local variables

Account: string

CUIDIndicator : Boolean

\*/

CUIDIndicator = False

/\*Check whether debit/credit indicator or a CHIPS Universal Identifier are present and delete depending on the case indicator and/or slash or double slash. \*/

**IF** **Substring**(MTPartyIdentifier, 1, 4) = “//CH” THEN

Account= **Substring**(MTPartyIdentifier, 5)

CUIDIndicator = True

**IF** Length(Account)= 0

**T11013** **~~T20233~~**

**ENDIF**

**ELSEIF** **Substring**(MTPartyIdentifier, 1, 3) = “/C/” OR “/D/” THEN

Account= **Substring**(MTPartyIdentifier, 4)

**T11008**

**ELSEIF Substring**(MTPartyIdentifier, 1, 2) = “/C” OR “/D” AND **Length**(MTPartyIdentifier, 3) = 0 THEN

/\* case where only the debit/credit indicator is used without account identification. \*/

Account= **“”**

**T11008**

**ELSE**

**Account = Substring**(MTPartyIdenfier, 2)

**ENDIF**

**IF** **IsIBAN**(Account)

Identification.IBAN = Account

**ELSE**

**IF LENGTH**(Account)> 0 THEN

Identification.Other.Identification = Account

**IF** CUIDIndicator = True

Identification.Other.SchemeName.Code = “CUID”

**ENDIF**

**ENDIF**

**ENDIF**

**Example 1 MT Source:**

:53B:/D/123456789

**MX Translation:**

<SttlmAcct>

<Id>

<Othr>

<Id>123456789</Id>

</Othr>

</Id>

</SttlmAcct>

**Example 2 MT Source:**

:52A:/BE43063816377701

*GKCCBEBB*

*(Italic not illustrated by this function)*

**MX Translation:**

<DbtrAgtAcct>

<Id>

<IBAN>BE43063816377701</IBAN>

</Id>

</DbtrAgtAcct>

**Example 3 MT Source:**

:58D://CH123456

*Bank ABC*

*US*

*(Italic not illustrated by this function)*

**MX Translation:**

<CdtrAcct>

<Id>

<Othr>

<Id>123456</Id>

<SchmeNm>

<Cd>CUID</Cd>

</SchmeNm>

</Othr>

</Id>

</CdtrAcct>

### 3.2.8 MT\_To\_MXIntermediaryAgent

**Business description**

The function extracts from MTAgent the information according to the pattern defined in MX\_To\_MT72FullField(2) and fill in the information in the MXAgent.

**Name**

MT\_To\_MXIntermediaryAgent

**Format**

**MT\_To\_MXIntermediaryAgent**(MTAgent ;MXIntermediaryAgent )

**Input** MTAgent (string)

**Output** MXIntermediaryAgent typed BranchAndFinancialInstitutionIdentification6

**Preconditions**

None

**Formal description**

The information in MTAgent is expected to be either

BIC OR a pattern like Name[**(**Country**(**TownName]OR ClearingSystemMemberIdentification

The way the “Name“ pattern has been built in MX to MT is described in MX\_To\_MTAgent.

The ClearingSystemMemberIdentification is the combination of ClearingSystemCode (5 char) defined in ISO code list and the MemberIdentification with max 28 char.

The ISO Clearing System Identification has not been converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

/\* Local Variables

MXName, MXCountry, MXTownName : string \*/

**IF** MTAgent **IsValidBIC** THEN

/\* For developers only - IsValidBIC must be checked versus a table like SWIFTREF with up-to-date information \*/

MXAgent.FinancialInstitutionIdentification.BICFI = MTAgent

**ELSEIF**

**IF WithinList**(Substring(MTAgent, 1, 5), ISOClearingSystemCode)

THEN

MXAgent. FinancialInstitutionIdentification.ClearingSystemMemberIdentification.ClearingSystemIdentification.Code = **Substring**(MTAgent, 1, 5)

**IF** **LENGTH**(Substring(MTAgent, 6) > 28 THEN

MXAgent. FinancialInstitutionIdentification.ClearingSystemMemberIdentification.MemberIdentification = **Concatenate**(**Substring**(MTAgent, 6, 27), “+”)

**ELSE**

MXAgent. FinancialInstitutionIdentification.ClearingSystemMemberIdentification.MemberIdentification = **Substring**(MTAgent, 6)

**ENDIF**

**ELSE**

MXName = **ExtractTillPattern**(MTAgent, ”(“ ISOCountryCode ”(“ )

/\* remove MXName from MTAgent \*/

MTAgent = **DeletePattern**(MTAgent, MXName)

**IF** **LENGTH**(MTAgent) > 0 THEN

MXTownName = **ExtractFromPattern**(MTAgent, 2, “(“)

/\* If FromPattern is not found, MXTownName is empty \*/

/\* Pattern is excluded \*/

MTAgent = **DeletePattern**(MTAgent, MXTownName)

**IF** **Substring**(MTAgent,2,2) **IsCountryCode** THEN

MXCountryCode = **Substring**(MTAgent,2,2)

**ENDIF**

**ENDIF**

**ENDIF**

/\* For developers only, IsCountryCode must be checked versus a list of country code maintained externally like SWIFT Reference Data \*/

/\* Restrict the length to comply with MX length. Just in case /INTA/ is created in a payment originated in MT which is unlikely \*/

**IF Length**(MXName)> 140 THEN

MXName = **Concatenate**(**Substring**(MXName,1,139), “+”)

**ENDIF**

**IF Length**(MXTownName)> 35 THEN

MXTownName = **Concatenate**(**Substring**(MXTownName,1,34), “+”)

**ENDIF**

**IF** **Length**(MXName) > 0 AND **Length**(MXTownName) > 0 AND **Length** (MXCountryCode) > 0 THEN

MXIntermediaryAgent.FinancialInstitutionIdentification.Name = MXName

MXIntermediaryAgent.FinancialInstitutionIdentification.PostaAddress .TownName = MXTownName

MXIntermediaryAgent.FinancialInstitutionIdentification.PostaAddress.Country = MXCountryCode

**ELSEIF LENGTH**(MXName) > 0

MXIntermediaryAgent.FinancialInstitutionIdentification.Name = MXName

MXIntermediaryAgent.FinancialInstitutionIdentification.PostaAddress.AddressLine = “NOTPROVIDED”

T11004 /\* Error code described in the error code list \*/

**ENDIF**

\*ISOClearingSystemCode is defined on [ISO 20022 Site](https://www.iso20022.org/external_code_list.page), [External Code Sets spreadsheet](https://www.iso20022.org/sites/default/files/documents/External_code_lists/ExternalCodeSets_1Q2019_May2019_v1.xls), sheet 5-ClearingSystemIdentification

## 3.3 Other Translation Rule Descriptions

The translation rule descriptions provided in this section are for translation rules that relate to any other fields or elements that are not specifically related to customer parties or financial institutions.

### 3.3.1 MT\_To\_MXRate

**Name**

MT\_To\_MXRate

**Business description**

The function translates an MT (exchange) rate to an MX (exchange) rate by replacing the decimal separator “,” by a “.”. The MX rate will be a decimal. Insignificant zeroes -if any- are deleted.

**Format**

**MT\_To\_MXRate**(MTRate ; MXRate)

**Input**

MTRate: (exchange) rate in the MT message expressed as 12d (up to 12 characters including the mandatory decimal separator “,”)

**Output**

MXRate: (exchange) rate in the MX message typed *BaseOneRate* with maximum 11 digits, maximum 10 fraction digits and a decimal separator “.”

**Preconditions** None.

**Formal description**

/\*Rate is a local variable\*/

/\*Replace the decimal separator “,”\*/

Rate = **ReplacePattern**(MTRate, “,”, “.”)

/\*Delete insignificant zeroes at the left and right\*/

Rate = **TrimLeft**(Rate, “0”)

Rate = **TrimRight**(Rate, “0”)

MXRate = Rate

**Examples**

MT Source: 0,1245

MX Translation: .1245 or 0.1245

MT Source: 1,21

MX Translation: 1.21

MT Source: 10,00

MX Translation: 10.0

MT Source: 1,

MX Translation: 1.0

### 3.3.2 MT\_To\_MXDate

**Name**

MT\_To\_MXDate

**Business description**

The function translates an MT date expressed as [YY]YYMMDD to an MX date “YYYY-MM-DD” by adding the century if absent and introducing the "-" separator between year, month and date.

**Format**

**MT\_To\_MXDate**(MTDate ; MXDate)

**Input**

MTDate: date in the MT message expressed as [YY]YYMMDD where

[YY] represents the century (optional) and YYMMDD represents the year, month and day

**Output**

MXDate: date in the MX message typed *ISODate* expressed as “YYYY-MM-DD”

**Preconditions**

None.

**Formal description**

/\*Check whether century is present and add if not. Year, Month

and Day are local variables\*/

**IF** **Length**(MTDate) = 8 THEN

/\*Date is of YYYYMMDD format\*/

Year = **Substring**(MTDate, 1, 4)

Month = **Substring**(MTDate, 5, 2)

Day = **Substring**(MTDate, 7, 2)

**ELSE**

/\*Date is of YYMMDD format\*/

Year = **Substring**(MTDate, 1, 2)

Month = **Substring**(MTDate, 3, 2)

Day = **Substring**(MTDate, 5, 2)

/\*Add century to Year\*/

Year = **Concatenate**(“20”, Year)

**ENDIF**

/\*Introduction pattern\*/

MXDate = **Concatenate**(Year, “-“, Month, “-“, Day)

**Examples**

|  |  |
| --- | --- |
| MT Source:  MX Translation: | 070101  2007-01-01 |
|  |  |
| MT Source:  MX Translation: | 20050101  2005-01-01 |

### 3.3.3 MT\_To\_MXTime

**Name**

MT\_To\_MXTime

**Business description**

The function translates an MT time expressed as “HHMM[SS[,3n]]” to an MX time expressed as “HH:MM:SS[.nnn...]” by:

* adding default seconds (“00”) to the MT time if absent
* introducing the “:“ pattern to separate hours, minutes and seconds
* replacing the “,” fraction of seconds separator by “.” if present

**Format**

**MT\_To\_MXTime**(MTTime ; MXTime)

**Input**

MTTime: time in the MT message expressed as “HHMM[SS][,3n]” where

HHMM represents hours and minutes

SS represents seconds (optional)

3n represents fraction of seconds (optional)

**Output**

MXTime: time in the MX message typed *ISOTime*, expressed as “HH:MM:SS[.nnn...]” where

HH:MM:SS represents hours, minutes and seconds nnn... represents fraction of seconds (optional)

**Preconditions**

None.

**Formal description**

/\*Hours, Minutes, Seconds and FractionS are local variables\*/

Hours = **Substring**(MTTime, 1, 2)

Minutes = **Substring**(MTTime, 3, 2)

/\*Check whether seconds and fraction of seconds are present\*/

**IF** **Length**(MTTime) = 4 THEN

/\*Time is of HHMM format. Assign default seconds SS\*/

Seconds = “00”

/\*Assign empty fraction of seconds string\*/ FractionS = “”

**ELSEIF** **Length**(MTTime) = 6

/\*Time is of HHMMSS format\*/

Seconds = **Substring**(MTTime, 5, 2)

/\*Assign empty fraction of seconds string\*/ FractionS = “”

**ELSE**

/\*Time is of HHMMSS,3n format\*/

Seconds = **Substring**(MTTime, 5, 2)

/\*Isolate fraction of seconds and replace fraction separator “,” by “.”\*/

FractionS = **Concatenate**(“.”, **Substring**(MTTime, 8))

**ENDIF**

/\*Introduction separator “:”\*/

MXTime = **Concatenate**(Hours, “:”, Minutes, “:”, Seconds,

FractionS)

**Examples**

MT Source: 1655

MX Translation: 16:55:00

MT Source: 165533

MX Translation: 16:55:33

MT Source: 165533,222

MX Translation: 16:55:33.222

### 3.3.4 MT\_To\_MXOffset

**Name**

MT\_To\_MXOffset

**Business description**

The function translates an MT offset time expressed as “HH[MM]” to an MX offset time expressed as “HH:MM” by:

adding default minutes (“00”) to the MT offset if absent introducing the “:“pattern to separate hours and minutes

**Format**

**MT\_To\_MXOffset**(MTOffset ; MXOffset)

**Input**

MTOffset: offset in the MT message expressed as “HH[MM]” where

HH represents hours and MM (optional) minutes

**Output**

MXOffset: offset in the MX message expressed as “HH:MM” where HH:MM represents hours and minutes

**Preconditions**

None.

**Formal description**

/\*Hours and Minutes are local variables\*/

Hours = **Substring**(MTOffset, 1, 2)

/\*Check whether optional minutes are present\*/

**IF** **Length**(MTOffset) = 4 THEN

/\*Offset is of HHMM format\*/

Minutes = **Substring**(MTOffset, 3, 2)

**ELSE**

/\*Offset is of HH format. Assign default minutes\*/

Minutes = “00”

**ENDIF**

/\*Introduction separator “:”\*/

MXOffset = **Concatenate**(Hours, “:”, Minutes) **Examples**

MT Source: 01

MX Translation: 01:00

MT Source: 0130

MX Translation: 01:30

### 3.3.5 MT\_To\_MXTimeOffset

**Name**

MT\_To\_MXTimeOffset

**Business description**

The function translates an MT time with offset expressed as “HHMM+/-HHMM” to an MX time with offset expressed as “HH:MM:SS+/-HH:MM” by adding the seconds to the time and introducing the “:“pattern to both time and offset.

**Format**

**MT\_To\_MXTimeOffset**(MTTimeOffset; MXTimeOffset)

**Input**

MTTimeOffset is made of :

MTCode is ignored in this function, if present

MTTime: time in the MT message expressed as “HHMM” (hours and minutes)

MTSign: offset sign in the MT message expressed as + or -

MTOffset: offset time in the MT message expressed as “HHMM” (hours and minutes)

**Output**

MXTimeOffset: time and offset in the MX message type *ISOTime*, expressed as

“HH:MM:SS+/-HH:MM” where

HH:MM:SS is the time

+ or - is the offset sign

HH:MM is the offset time

**Preconditions**

None.

**Formal description**

/\*The function calls sub-functions *MT\_To\_MXTime* and

*MT\_To\_MXOffset* to translate the time formats. MXTime and

MXOffset are local variables\*/

MXTime = **MT\_To\_MXTime(MTTime)**

MXOffset = **MT\_To\_MXOffset(MTOffset)**

/\*Build output string\*/

MXTimeOffset = **Concatenate**(MXTime, MTSign, MXOffset)

**Example**

MT Source: 1355+0100

MX Translation: 13:55:00+01:00

### 3.3.6 MT\_To\_MXDateTimeOffset

**Name**

MT\_To\_MXDateTimeOffset

**Business description**

The function translates an MT date expressed as [YY]YYMMDD and an MT time with offset expressed as “HHMM+/-HHMM” to an MX date and time with offset expressed as “YYYY-MM-DDTHH:MM:SS+/-HH:MM” by:

* adding the century if absent and introducing the “-“ separator between year, month and date
* introducing the “T” separator between date and time with offset
* adding the seconds to the time and introducing the “:“separator between hours, minutes and seconds
* adding the separator “:” between hours and minutes to the source offset.

**Format**

**MT\_To\_MXDateTimeOffset**(MTDate, MTTime, MTSign, MTOffset ; MXDateTime)

**Input**

MTDate: date in the MT message expressed as “[YY]YYMMDD” where [YY] represents the optional centure and YYMMDD the year, month and day

MTTime: time in the MT message expressed as “HHMM” (hours and minutes)

MTSign: offset sign in the MT message expressed as + or -

MTOffset: offset time in the MT expressed as “HHMM” (hours and minutes) **Output**

MXDateTime: date and time in the MX message typed *ISODateTime* expressed as

“YYYY-MM-DDTHH:MM:SS+/-HH:MM where

YYYY-MM-DD is the date

T is the date/time separator

HH:MM:SS is the time

+/- is the offset sign

HH:MM is the offset time

**Preconditions**

None.

**Formal description**

/\*The function calls sub-functions *MT\_To\_MXDate* and *MT\_To\_MXTimeOffset* to translate the date and time with offset formats. MXDate and MXTimeOffset are local variables\*/

MXDate = **MT\_To\_MXDate**(MTDate)

MXTimeOffset = **MT\_To\_MXTimeOffset**(MTTime, MTSign, MTOffset)

/\*Build output string\*/

MXDateTime = **Concatenate**(MXDate, “T”, MXTimeOffset)

**Examples MT Source:**

Time with offset 1355+0100

Date 070222

**MX Translation:** 2007-02-22T13:55:00+01:00

### 3.3.7 MT\_To\_MXCurrencyAmount

**Name**

MT\_To\_MXCurrencyAmount

**Business description**

The function translates an MT currency and MT amount to an MX amount with the currency embedded as an XML attribute. The decimal separator “,” is replaced by a “.”. If the MT amount ends with a “,” then decimal separator is dropped and the translated MX amount will be an integer. Insignificant zeroes -if any- are deleted.

This is needed to avoid a situation where JPY12345, is translated to 12345.0 JPY which is invalid as the JPY currency does not allow decimal digits.

**Format**

**MT\_To\_MXCurrencyAmount**(MTCurrency, MTAmount ; MXAmount)

**Input**

MTCurrency: currency in the MT message expressed as 3!a (exactly 3 alphabetic capital letters)

MTAmount: amount in the MT message expressed as 15d (up to 15 characters including the mandatory decimal separator “,”)

**Output**

MXAmount: amount in the MX message typed *ActiveOrHistoricCurrencyAndAmount or ActiveCurrencyAndAmount* with maximum 18 digits, maximum 5 fraction digits and minimum inclusive amount is zero. CBPR+ reduces the number of total digits to 14. The decimal separator “.” is optional in XML if the amount is an integer. The currency is embedded as and XML attribute.

**Preconditions**

It is assumed that both MT and MX currencies are either active or active and historic. If it is not the case, validation problem will be expected when the target is more constrained.

The same level of validation on MT and MX is requested to check pairs of (currency, amount) in terms of allowed decimals number.

**Formal description**

/\*Replace decimal separator “,” by “.”. Amount is a local variable\*/

Amount = **ReplacePattern**(Amount, “,”, “.”)

/\*Delete insignificant zeroes at the left and right\*/

Amount = **TrimLeft**(Amount, “0”)

Amount = **TrimRight**(Amount, “0”)

/\*Check whether the amount has a fractional part. If not, the decimal separator is deleted. Position is a local variable and indicates the position of the last character\*/

Position = **Length**(Amount)

**IF** **Substring**(Amount, Position) = “.”

MXAmount = **DeletePattern**(Amount, “.”)

**ELSE**

MXAmount = Amount

**ENDIF** /\*Define currency attribute\*/

MXAmount.XMLAttribute(Ccy) = MTCurrency

**Examples**

|  |  |
| --- | --- |
| MT Source:  MX Translation: | USD1548,00  <Amt Ccy = "USD">1548</Amt> |
| MT Source:  MX Translation: | USD1548,50  <Amt Ccy = "USD">1548.5</Amt> |
| MT Source:  MX Translation: | USD1548,33  <Amt Ccy = "USD">1548.33</Amt> |
|  |  |

### 3.3.8 MT70ROC\_To\_MX35Text

**Name**

MT70ROC\_To\_MX35Text

**Business description**

The function isolates the string that follows a code /ROC/ in an MT remittance information field 70 and restricts the output string to maximum 35 characters.

**Format**

**MT70ROC\_To\_MX35Text**(MT70 ; MX35Text)

**Input**

MT70: remittance information in an MT field 70 (format 4\*35x).

**Output**

MX35Text: string of characters in the MX message typed *Max35Text*.

**Preconditions**

None.

**Formal description**

/\*MT70 is defined by 4 lines of 35 characters\*/

/\*Basic function *ExtractBetweenPattern* extracts the information between the code /ROC/ and one of the other available codes preceded by // in field 70: ///INV/, ///RFB/, ///IPI/ and ///TSU/. From a previous MX to MT translation, other code words could be present like /ULTB/, /ULTD/, /PURP/, /URI/, /RELID/, /SRI/. In order to be generic, the search of one of these code words will be limited to the search of “///”.IF “///” is notencountered, the output string will be the information following /ROC/ until the end of field 70. If the information is spread over more than one line, then the function removes the MT line separator "*CRLF*". If the length of the information is more than 35 char, the first 34 characters are extracted followed by “+” to indicate truncation. MT70ROC is a local variable\*/

MT70ROC = **ExtractBetweenPattern**(MT70, “/ROC/”,”///”)

/\*Limit the output string to 35 characters\*/

**IF** LENGTH(MT70ROC) > 35 THEN

MX35Text = **Concatenate**(**Substring**(MT70ROC, 1, 34), “+”)

**ELSE**

MX35Text = Substring(MT70ROC,1,35)

**ENDIF**

**Example 1 MT Source:**

:70:/ROC/CUST123456789///INV/123 dated 03/04/2006

**MX Translation (to EndToEndId):**

<PmtId>

<EndToEndId>CUST123456789</EndToEndId>

</PmtId>

**Example 2 MT Source:**

:70:/INV/Invoice 123 dated 03/04/2006///ROC/A

C-123456789C

**MX Translation (to EndToEndId):**

<PmtId>

<EndToEndId>AC-123456789C</EndToEndId>

</PmtId>

### 3.3.9 MT\_To\_MXRemittanceInformation

**Name**

MT\_To\_MXRemittanceInformation

**Business description**

The function translates an MT remittance information field 70 to an MX unstructured remittance information element. When 2 consecutive lines are concatenated a space is added between them when the first line is not completely filled in. Otherwise it is interpreted as the second line being the continuation of the first line where a word could even be split between the 2 lines.

**Format**

**MT\_To\_MXRemittanceInformation**(MT70 ; MXRemittanceInformation)

**Input**

MT70: remittance information in an MT field 70 (format 4\*35x).

**Output**

MXRemittanceInformation: remittance information in the MX message typed *Max140Text*.

**Preconditions**

None.

**Formal description**

/\*MT70 is defined by 4 lines of 35 characters. MT70[1]indicates the first line, not including the Carriage Return Line Feed “*CRLF*” separator needed between consecutive lines in an MT field with multiple lines\*/

/\*Concatenate all lines in one string. When 2 consecutive lines are concatenated a space is added between them when the first line is not completely filled in \*/

/\* Local variable

SPACE : string

i : integer \*/

SPACE = “SPACE” character value

**IF** **Length**(MT70)> 0 THEN

MXRemittanceInformation= MT70[1]

**ENDIF**

**For i** = 2 to 4

**IF** **Length**(MT70[i])> 0 THEN

**IF** **Length**(MT70[i-1]) < 35 THEN

MXRemittanceInformation = **Concatenate**(MXRemittanceInformation, SPACE, MT70[i])

**ELSE**

MXRemittanceInformation = **Concatenate**(MXRemittanceInformation,MT70[i])

**ENDIF**

**ENDIF**

**Next i**

**Example MT Source:**

:70:/ROC/CUST123456789///INV/Invoice 12

3 dated 03/04/2006

**MX Translation (to RemittanceInformation\Unstructured):**

<RmtInf>

<Ustrd>/ROC/CUST123456789///INV/Invoice 123 dated 03/04/2006</Ustrd>

</RmtInf>

### 

### 3.3.10 MT72\_To\_MXText

**Name**

MT72\_To\_MXText

**Business description**

The function isolates the information following a code (passed as input parameter of the function) in an MT field 72 and restricts the MX output string to a maximum length (passed as input parameter of the function). The code is deleted in the returned information. If the length of the extracted information without the code is longer than the length parameter, sign “+” is returned in the last character to indicate truncation of the information.

**Format**

**MT72\_To\_MXText**(MT72, MTCode, Length ; MXText)

**Input**

MT72: content of an MT field 72 with format 6\*35x (6 lines of 35 characters). MTCode: code as used in an MT field 72 (for example /REC/, /ACC/, /BNF/…).

Length: maximum length of the output string.

**Output**

MXText: MX string of characters.

**Preconditions** None.

**Formal description**

/\*Field 72 is defined by 6 lines of 35 characters with “//” as line continuation pattern and MT line separator “*CRLF*” between consecutive lines\*/

/\*Basic function *ExtractLines* extracts the information with the

MTCode. If the information is spread over more than 1 line, then the function removes the MT line separator "*CRLF*" and replaces the continuation pattern "//" by a space “ ”.

MT72Extract is a local variable\*/

MT72Extract = **ExtractLines**(MT72, MTCode, “//”)

/\*Substring is taken to delete the MTCode at the start of the output string and to restrict the length as per the maximum passed with the input parameter Length. If the length is greater than the length parameter, the last character is replaced by “+” to indicate truncation. Position is a local variable to indicate where to start the substring in the MTExtract \*/

Position = **Length**(MTCode) + 1

MXText **= Substring(MT72Extract, Position)**

**IF** **Length**(MXText) > Length THEN

MXText= **Concatenate**(**Substring**(MXText, 1, Length -1), “+”)

**ENDIF**

**Example MT Source:**

:72:/ACC/Payment for further credit

//to subsidiary ABC, US, Miami

**MX Translation (to InstructionForCreditorAgent\InstructionInformation element):**

<InstrForCdtrAgt>

<InstrInf>Payment for further credit to subsidiary ABC, US, Miami</InstrInf>

</InstrForCdtrAgt>

### 3.3.11 MT\_To\_ MXRegulatoryReporting

**Name**

MT\_To\_MXRegulatoryReporting

**Business description**

Field 77B has the format : Narrative (3\*35x) OR

Line 1 : /8a/2!a[//AdditionalInformation]

Line 2-3 : [//continuation of additional information]

Where /8a/ is one of the code /BENEFRES/ or /ORDERRES/

2!a is the country code of the Residence of the Beneficiary/Creditor or Debtor

From customer samples, it is seen that the 2 code words can be used in field 77B and even a mix of codes and non codes as shown in the examples below. When the continuation sign is not used (Example 1), it is difficult to interpret if the second line is the continuation of the first line or another information. That is why the full information will be copied to RegulatoryReporting with the possibility to have information duplication in case only a code word is used with only the country of Residence as shown in Example 2

Example 1

:77B:/ORDERRES/BE//MEILAAN 1, 9000 GENTT

77245678901234567890123456789012345

77345678901234567890123456789012345

Example 2

:77B:/ORDERRES/FR

/BENEFRES/EG//INVOICE SETTLEMENT

The function searches for the code word /BENEFRES/ and /ORDERRES/ and extracts the CountryOfResidence. If it is a valid country code, it is translated to the Creditor (or Debtor)/CountryOfResidence.

The Field 77B is then copied as such to RegulatoryReporting[1].Details[1].Information[n]

**Format**

**MT\_To\_MXRegulatoryReporting**(MTRegulatoryReporting ; MXRegulatoryReporting; MXDebtor, MXCreditor)

**Input**

MTRegulatoryReporting: regulatory reporting details in an MT field 77B (format 3\*35x).

**Output**

MXRegulatoryReporting: MX element of one or more occurrences of the target component typed *RegulatoryReporting3*.

MXDebtor, MXCreditor are typed PartyIdentification135

**Preconditions**

None.

**Formal description**

/\*MTRegulatoryReporting is defined by 3 lines of 35 characters.

MTRegulatoryReporting[1] indicates the first line not including the Carriage Return Line Feed “*CRLF*” needed as separator

between consecutive lines in an MT field with multiple lines format\*/

/\* Local variables

CreditorLine, DebtorLine, CountryCode : string

CreditorResPattern, DebtorResPattern : pattern \*/

CreditorResPattern = “/BENEFRES/” followed by exactly 2 alphabetic char, upper case (which should be a Country code)

DebtorResPattern = “/ORDERRES/” followed by exactly 2 alphabetic char, upper case (which should be a Country code)

/\* The 2 codes can be present as per customer samples \*/

CreditorLine = **ExtractPattern**(MTRegulatoryReporting, CreditorResPattern)

DebtorLine = **ExtractPattern**(MTRegulatoryReporting, DebtorResPattern)

/\* Fill in the Party Country Of Residence \*/

**IF** **Length**(CreditorLine) > 0 THEN

CountryCode = Substring(CreditorLine,11,2)

**IF** **IsCountryCode**(CountryCode) THEN

MXCreditor.CountryOfResidence = CountryCode

**ENDIF**

**ENDIF**

**IF** **Length**(DebtorLine) > 0 THEN

CountryCode = Substring(DebtorLine,11,2)

**IF** IsCountryCode(CountryCode) THEN

MXDebtor.CountryOfResidence = CountryCode

**ENDIF**

**ENDIF**

/\* Copy 3\*35 char from Field 77B to RegulatortReporting \*/

{

/\*The function translates each MTRegulatoryReporting line to the Information element (Max35Text) in a different occurrence of the RegulatoryReporting[1].Details[1].Information.

RegulatoryReporting[1].Details[1].Information[1] indicates

the first occurrence of the Information \*/

/\*Translation first line\*/

RegulatoryReporting[1].Details[1].Information[1] =

MTRegulatoryReporting[1]

/\*Check presence of further lines and possible translation\*/

**IF** **Length**(MTRegulatoryReporting[2]) > 0 THEN RegulatoryReporting[1].Details[1].Information[2] =

MTRegulatoryReporting[2]

**IF** **Length**(MTRegulatoryReporting[3]) > 0 THEN RegulatoryReporting[1]Details[1].Information [3] =

MTRegulatoryReporting[3]

**ENDIF**

**ENDIF**

}

**Example MT Source:**

:77B:/ORDERRES/BE

//MEILAAN 1, 9000 GENTT

/BENEFRES/US

**MX Translation:**

<Dbtr>

<CtryOfRes>BE</CtryOfRes>

</Dbtr>

<Cdtr>

<CtryOfRes>US</CtryOfRes>

</Cdtr>

<RgltryRptg>

<Dtls>

<Inf>/ORDERRES/BE</Inf>

<Inf>//MEILAAN 1, 9000 GENTT</Inf>

<Inf>/BENEFRES/US</Inf>

</Dtls>

</RgltryRptg>

### 3.3.12 MT\_To\_ MXInstruction

**Name**

MT\_To\_MXInstruction

**Business description**

This function translates MT 101, field 23E instructions CMSW, CMTO, CMZB, URGP and (the first occurrence of) OTHR by combining them in a string of MX instruction(s).

**Format**

**MT\_To\_MXInstruction** (MT23E,... ; MXInstruction)

**Input**

MT23E: field 23E composed of the mandatory subfield MTInstructionCode (for example CMTO, CMSW, OTHR...) and optional MTInstructionInfo (code dependent). As field 23E is repetitive, all occurrences of field 23E are passed as input variables.

**Output**

MXInstruction: string of characters representing the MX instruction(s).

**Preconditions**

None.

**Formal description**

/\*MTInstructionInfo[CODE] indicates to which CODE the information is related. If a code can be repeated, then MTInstructionInfo[CODE, N] will indicate the information with

the Nth occurrence of that CODE. For example,

MTInstructionInfo[OTHR, 1] contains the information that comes with the first occurrence of OTHR\*/

/\*Check whether OTHR occurs and whether MTInstructionInfo[OTHR] is not empty. If there is no information, the code OTHR is not translated\*/

**IF** MTInstructionCode = “OTHR” AND

**Length(**MTInstructionInfo[OTHR, 1])>0

/\*Check whether URGP occurs. URGP can be combined with any of the other codes that this functions considers for translation\*/

**IF** MTInstructionCode = “URGP”

**IF** MTInstructionCode = “CMSW”

MXInstruction = **Concatenate**(“CMSW/URGP/OTHR/”,

MTInstructionInfo[OTHR, 1])

**ELSEIF** MTInstructionCode = “CMTO”

/\*With CMTO, additional information is allowed.

CMTO cannot be repeated\*/ **IF Length(**MTInstructionInfo[CMTO])> 0 THEN

MXInstruction = **Concatenate**(“CMTO/”,

MTInstructionInfo[CMTO], “/URGP/OTHR/”, MTInstructionInfo[OTHR, 1])

**ELSE**

MXInstruction =

C**oncatenate**(“CMTO/URGP/OTHR/”,

MTInstructionInfo[OTHR, 1])

**ENDIF**

**ELSEIF** MTInstructionCode = “CMZB”

MXInstruction = **Concatenate**(“CMZB/URGP/OTHR/”, MTInstructionInfo[OTHR, 1])

**ELSE**

MXInstruction = **Concatenate**(“URGP/OTHR/”, MTInstructionInfo[OTHR, 1])

**ENDIF**

**ELSEIF** MTInstructionCode = “CMSW”

MXInstruction = **Concatenate**(“CMSW/OTHR/”,

MTInstructionInfo[OTHR, 1])

**ELSEIF** MTInstructionCode = “CMTO”

/\*With CMTO, additional information is allowed.

CMTO cannot be repeated\*/

**IF** **Length(**MTInstructionInfo[CMTO])> 0

MXInstruction = **Concatenate**(“CMTO/”,

MTInstructionInfo[CMTO], “/OTHR/”, MTInstructionInfo[OTHR, 1])

**ELSE**

MXInstruction = **Concatenate**(“CMTO/OTHR/”, MTInstructionInfo[OTHR, 1])

ENDIF

**ELSEIF** MTInstructionCode = “CMZB”

MXInstruction = **Concatenate**(“CMZB/OTHR/”,

MTInstructionInfo[OTHR, 1])

**ELSE**

MXInstruction = **Concatenate**(“OTHR/”,

MTInstructionInfo[OTHR, 1])

**ENDIF**

/\*OTHR is not present, same check for other codes as above is done\*/

**ELSEIF** MTInstructionCode = “URGP”

**IF** MTInstructionCode = “CMSW” MXInstruction = “CMSW/URGP”

**ELSEIF** MTInstructionCode = “CMTO”

/\*With CMTO, additional information is allowed.

CMTO cannot be repeated\*/

**IF** **Length(**MTInstructionInfo[CMTO])> 0

MXInstruction = **Concatenate**(“CMTO/”, MTInstructionInfo[CMTO], “/URGP”)

**ELSE**

MXInstruction = “CMTO/URGP”

**ENDIF**

ELSEIF MTInstructionCode = “CMZB”

MXInstruction = “CMZB/URGP”

**ELSE**

MXInstruction = “URGP”

**ENDIF**

**ELSEIF** MTInstructionCode = “CMSW”

MXInstruction = “CMSW”

**ELSEIF** MTInstructionCode = “CMTO”

/\*With CMTO, additional information is allowed. CMTO cannot be repeated\*/

**IF** **Length(**MTInstructionInfo[CMTO])> 0

MXInstruction = **Concatenate**(“CMTO/”,

MTInstructionInfo[CMTO])

**ELSE**

MXInstruction = “CMTO”

**ENDIF**

**ELSEIF** MTInstructionCode = “CMZB” MXInstruction = “CMZB”

**ENDIF**

**Example**

**MT Source:**

:23E: CMSW

:23E: OTHR/Service 1

**MX Translation:**

MX InstructionForDebtorAgent

<InstrForDbtrAgt>CMSW/OTHR/Service 1</InstrForDbtrAgt>

### 3.3.13 MTDate4\_To\_MXDate

**Name**

MTDate4\_To\_MXDate

**Business description**

The function translates an MT date expressed as MMDD to an MX date “YYYY-MM-DD” by:

* Copying the [YY]YY from the second MT date parameter
* adding the century if absent
* introducing the "-" separator between year, month and date.

**Format**

**MTDate4\_To\_MXDate**(MTDate4, MTDate; MXDate)

**Input**

MTDate4: date in the MT message expressed as MMDD where MMDD represents the month and day

MTDate: date in the MT message expressed as [YY]YYMMDD where [YY]: represents the century (optional) and YYMMDD represents the year, month and day

**Output**

MXDate: date in the MX message typed *ISODate* expressed as “YYYY-MM-DD” **Preconditions**

None.

**Formal description**

/\*Extract Month4 and Day4 from the first MT date parameter.

Month4 and Day4 are both local variables. \*/

Month4 = **Substring**(MTDate4, 1, 2)

Day4 = **Substring**(MTDate4, 3, 2)

/\*Check whether the century is present and, if not, add it.

Year is a local variable\*/

**IF** **Length**(MTDate) = 8

/\*Date is of YYYYMMDD format\*/

Year = **Substring**(MTDate, 1, 4)

**ELSE**

/\*Date is of YYMMDD format\*/

Year = **Substring**(MTDate, 1, 2)

/\*Add century to Year\*/

Year = **Concatenate**(“20”, Year)

**ENDIF**

/\*Introduction pattern\*/

MXDate = **Concatenate**(Year, “-“, Month4, “-“, Day4)

**Examples**

|  |  |
| --- | --- |
| MT Source:  MX Translation: | 0101 080103  2008-01-01 |
| MT Source:  MX Translation: | 0101 20080103  2008-01-01 |
| MT Source:  MX Translation: | 0101 20051229  2005-01-01 |
|  |  |

### 3.3.14 MT\_To\_MXSum

**Name**

MT\_To\_MXSum

**Business description**

The function translates an MT amount to an MX sum. The decimal separator “,” is replaced by a “.”. If the MT amount ends with a “,” then decimal separator is dropped and the translated MX amount will be an integer. Insignificant zeroes -if any- are deleted.

**Format**

**MT\_To\_MXSum**( MTAmount ; MXSum)

**Input**

MTAmount: amount in the MT message expressed as 15d (up to 12 characters including the mandatory decimal separator “,”)

**Output**

MXSum: sum in the MX message typed *DecimalNumber* with maximum 18 digits in total, maximum 17 fraction digits and an optional decimal separator “.”.

**Preconditions**

The same level of validation on MT and MX is requested to check pairs of currency and amount in terms of allowed decimals number, although the currency is not written in the MX output.

**Formal description**

/\*Replace decimal separator “,” by “.”. Amount is a local variable\*/

Amount = **ReplacePattern**(Amount, “,”, “.”)

/\*Delete insignificant zeroes at the left and right\*/

Amount = **TrimLeft**(Amount, “0”)

Amount = **TrimRight**(Amount, “0”)

/\*Check whether the amount has a fractional part. If not, the decimal separator is deleted. Position is a local variable and indicates the position of the last character\*/

Position = **Length**(Amount)

**IF** **Substring**(Amount, Position) = “.” THEN

MXSum = **DeletePattern**(Amount, “.”)

**ELSE**

MXSum = Amount

**ENDIF**

**Example 1 MT Source:**

:90D:*75475USD*123,52

**MX Translation**

*<TxsSummry>*

*<TtlDbtNtries>*

*<NbOfNtries>75475</NbOfNtries>*

<Sum>123.52</Sum>

*</TtlDbtNtries>*

*</TxsSummry>*

*(Italic not translated by this function)*

### 3.3.15 MT\_To\_MX InstructionForCreditorAgent

**Business description**

InstructionForCreditorAgent can be filled from field72 with code “/ACC/” or field 23E with codes “CHQB”, “HOLD”, “PHOB” and “TELB”. In order to optimise the mapping, the type of configuration is analysed first (source from both MT72 and MT23 or only source from MT72 or only from MT23E).

If information from Field 72 must be truncated the sign “+” is added as last character.

**Name**

MT\_To\_MXInstructionForCreditorAgent

**Format**

**MT\_To\_MXInstructionForCreditorAgent** (MT23E, MT72 ;MXInstructionForCreditorAgent )

**Input**

**Output** MXInstructionForCreditorAgent typed InstructionForCreditorAgent1

**Preconditions**

**Formal description**

/\* Local variables

ConfigIndicator, MXText : string

N, CodeNumber : integer

First is Boolean \*/

First = “true ”

CodeNumber = 0 /\* Count the number of codes.\*/

n **= NumberOfOccurrences**(23E)

**For j** = 1 to n

IF 23E[j] IsinList {HOLD, PHOB, CHQB,TELB} THEN

CodeNumber = CodeNumber +1

**j = j+1**

**IF** **IsPresentPattern**(MT72, "/ACC/") THEN

MT72\_To\_MXText(MT72,”/ACC/”,205;MXText)

**ENDIF**

**IF** **Length**(MXText) > 0 THEN

**IF** CodeNumber >0 THEN

ConfigIndicator = “ACCCode”

**ELSE**

ConfigIndicator = “ACCOnly”

**ENDIF**

**ELSE**

**IF** CodeNumber >0 THEN

ConfigIndicator = “CodeOnly”

**ELSE**

/\* Nothing to translate \*/

**EXIT Function**

**ENDIF**

**ENDIF**

/\* Fill in the element InstructionForCreditorAgent \*/

**Case** ConfigIndicator = “ACCOnly”

/\* The 2 occurrences of InstructionForCreditorAgent are filled in with information from field 72 As Field 72 is max 210 char, the max info after the code is less than 205 char. No truncation needed \*/

**MT72\_To\_MXText**(MT72,”/ACC/",205;MXText)

InstructionForCreditorAgent[1]. InstructionInformation = **Substring**(MXText, 1, 140)

InstructionFoCreditorAgent[2]. InstructionInformation = **Substring**(MXText, 141, 280)

**Case** ConfigIndicator = “ACCCode”

/\* First occurrence is filled in with information from Field 72 (without code); Truncation may be needed\*/

/\* Second occurrence of InstructionForCreditorAgent is filled in with information from field 23E with the following structure:

InstructionForCreditorAgent[2]. Code = Code1

InstructionForCreditorAgent[2]. InstructionInformation =[Code1 related information], [/Code2/ [Code2 related information]] [/Code3/ [Code3 related information]] [/Code4/[Code4 related information]] with a max possible length of 138 char . No truncation needed.

Where [ ] means optional\*/

**MT72\_To\_MXText**(MT72,"/ACC/",140;MXText)

InstructionForCreditorAgent[1]. InstructionInformation= MXText

/\* Fill in InstructionForCreditorAgent[2]. InstructionInformation \*/

**For j** = 1 to n

**IF** 23E[j] IsInList {HOLD, PHOB, CHQB,TELB} THEN

**IF** First = “True” Then

InstructionForCreditorAgent[2].Code = 23E[j].Instruction Code

InstructionForCreditorAgent[2]. InstructionInformation = **Substring**(23E[j].AdditionalInformation, 2, 30)

/\* starts at 2 to remove “/”. AdditionalInformation is optional. If Empty nothing is copied \*/

First = “False”

**ELSE**

InstructionForCreditorAgent[2]. InstructionInformation =

**Concatenate**(InstructionForCreditorAgent[2]. InstructionInformation,“/”,23E[j].Instruction Code, “/”,

**Substring**(23E[j].AdditionalInformation, 2, 30)

/\* starts at 2 to remove “/”. AdditionalInformation is optional. If Empty nothing is copied \*/

**ENDIF**

**ENDIF**

**j = j+1**

**Case** ConfigIndicator = “CodeOnly”

/\*

The following template is used:

InstructionForCreditorAgent[1]. Code = Code1

InstructionForCreditorAgent[1]. InstructionInformation = Code1 related information, if present

InstructionForCreditorAgent[2]. Code = Code2

InstructionForCreditorAgent[2]. InstructionInformation =[Code2 related information][/Code3/ [Code3 related information][/Code4/[Code4 related information]]

Where [ ] means optional as information might be absent. \*/

**For** j = 1 to n

**IF** 23E[j] IsInList {HOLD, PHOB, CHQB,TELB} THEN

**IF** First = “True” Then

InstructionForCreditorAgent[1].Code = 23E[j].Instruction Code

InstructionForCreditorAgent[1]. InstructionInformation = **Substring**(23E[j].AdditionalInformation, 2, 30)

/\* starts at 2 to remove “/”. AdditionalInformation is optional. If Empty nothing is copied \*/

First = “False”

**ELSEIF**

First = “False” AND InstructionForCreditorAgent[2].Code **IsEmpty** THEN

InstructionForCreditorAgent[2].Code = 23E[j].Instruction Code

InstructionForCreditorAgent[2]. InstructionInformation = **Substring**(23E[j].AdditionalInformation, 2, 30)

/\* starts at 2 to remove “/”. AdditionalInformation is optional. If Empty nothing is copied \*/

**ELSE**

/\* First = “False” And InstructionForCreditorAgent[2].Code Is not Empty \*/

InstructionForCreditorAgent[2]. InstructionInformation =

**Concatenate**(InstructionForCreditorAgent[2]. InstructionInformation,“/”,23E[j].Instruction Code, “/”,

**Substring**(23E[j].AdditionalInformation, 2, 30)

/\* starts at 2 to remove “/”. AdditionalInformation is optional. If Empty nothing is copied \*/

**ENDIF** /\* IF First = “True \*/

**ENDIF** /\* IF 23E[j] IsInList \*/

**j=j+1**

**END FOR**

### 3.3.16 MT\_To\_MX InstructionForCreditorAgent2

**Business description**

InstructionForCreditorAgent is filled from field72 information with code “/ACC/” or code “/PHONBEN/” or code “/TELEBEN/”. These last 2 codes have an ISO equivalent code respectively “”/PHOB/” and “/TELB/” while the other code /ACC/ has no ISO equivalent. In order to optimise the translation, the type of configuration is analysed first in order to better fill the InstructionForCreditorAgent limited to 2 occurrences and made of an ISO Code and a free text InstructionInformation of maximum 140 characters.

If information from Field 72 must be truncated the sign “+” is added as last character.

Additional cases to be considered as per CBPR+ WG change request V2.1 in pacs.009 CORE used to settle a previously sent pacs.009 ADV. A specific code /UDLC/ can be carried in InstructionForCreditorAgent.InstructionInformation (refer to SubfunctionInstructionForCreditorAgent). As a consequence, there are 2 additional cases to consider in the translation back to MX : /UDLC/ and /ACC/ both present and /UDLC/ is present without /ACC/. /UDLC/Info is expected to use max 1 occurrence of InstructionInformation otherwise it is truncated. If both /UDLC/ and /ACC/ are present and if /PHONBEN/ or /TELEBEN/ are also present, these last 2 codes are not translated. IF /ACC/ is absent and /UDLC/ is present there is still room left for /PHONBEN/ and/or /TELEBEN/ in the other occurrence of InstructionInformation

In any case, /UDLC/Info is translated to the last available occurrence of InstuctionInformation (see rationale in SubfunctionInstructionForCreditorAgent)

**Name**

MT\_To\_MXInstructionForCreditorAgent2

**Format**

**MT\_To\_MXInstructionForCreditorAgent2** (MT72 ;MXInstructionForCreditorAgent )

**Input**

**Output** MXInstructionForCreditorAgent typed InstructionForCreditorAgent2

**Preconditions**

**Formal description**

**/\* Local variables**

ConfigIndicator is a string

MXText, MXACCText, MXUDLCText is a string

MXInstructionInformation is a string

First is a Boolean

RemainingLength, RemainingLength1, RemainingLength2 : integer

\*/

First = “true”

/\* Extract information following the code \*/

**IF** **IsPresentPattern**(MT72, "/ACC/") THEN

**MT72\_To\_MXText**(MT72,”/ACC/”,205;MXACCText)

**ENDIF**

**IF** **IsPresentPattern**(MT72, "/UDLC/") THEN

**MT72\_To\_MXText**(MT72,”/UDLC/”,204;MXUDLCText)

**ENDIF**

**IF Length**(MXUDLCText)= 0 THEN

**{**

**IF** **Length**(MXACCText) > 0 THEN

**IF** **IsPresentPattern**(MT72, "/PHONBEN/") OR **IsPresentPattern**(MT72, "/TELEBEN/") THEN

ConfigIndicator = “ACCCode”

**ELSE**

ConfigIndicator = “ACCOnly”

**ENDIF**

**ELSE**

**IF** **IsPresentPattern**(MT72, "/PHONBEN/") OR **IsPresentPattern**(MT72, "/TELEBEN/") THEN

ConfigIndicator = “CodeOnly”

**ELSE**

/\* Nothing to translate \*/

EXIT Function

**ENDIF**

**ENDIF**

**}**

**ELSE /\* Length**(MXUDLCText)<> 0 \*/

**IF** **Length**(MXACCText) > 0 THEN

ConfigIndicator = “UDLCACC”

**ELSE**

ConfigIndicator = “UDLC”

**ENDIF**

**ENDIF /\* End IF Length**(MXUDLCText)= 0 \*/

/\* Fill in the element InstructionForCreditorAgent \*/

**Case** ConfigIndicator = “ACCOnly”

/\* This configuration means that only /ACC/ is present. \*/

/\* The 2 occurrences of InstructionForCreditorAgent can be filled in with information from field 72 /ACC/. As Field 72 is max 210 char, no truncation needed. Code /ACC/ is removed in MX. \*/

/\* Build the string\*/

**IF** **Length**(MXACCText) > 0 THEN

MXText = MXACCText

**ENDIF**

InstructionForCreditorAgent[1]. InstructionInformation = **Substring**(MXText, 1, 140)

InstructionFoCreditorAgent[2]. InstructionInformation = **Substring**(MXText, 141, 280)

**Case** ConfigIndicator = “ACCCode”

/\* This configuration means that (/ACC/) AND (/PHONBEN/ or /TELEBEN/) are present.\*/

/\* First occurrence is filled in with information from Field 72 (/ACC/ without ISO code). Truncation may be needed if information is longer than 140 char\*/

/\* Second occurrence of InstructionForCreditorAgent is filled in with information from /PHONBEN/ (ISO code is /PHOB/) or /TELEBEN/ (ISO code is /TELB/) with the following structure:

InstructionForCreditorAgent[2].Code = Code1

InstructionForCreditorAgent[2].InstructionInformation =[Code1 related information],[/Code2/ [Code2 related information]] with a max 140 char.Code1 and Code2 refer to ISO codes. It is unlikely that truncation will be needed. It is also unlikely that the 2 codes are used together as their meaning is rather exclusive. Nevertheless the function handles the cases where the 2 codes are present

Where [ ] means optionally present \*/

/\* Fill InstructionForCreditorAgent[1] \*/

**MT72\_To\_MXText**(MT72, "/ACC/", 140; MXText)

/\* MXText might be empty if /ACC/ is absent or /ACC/ is present but with no information following the code. This should not be the case at this point\*/

InstructionForCreditorAgent[1]. InstructionInformation= MXText

/\* Fill in InstructionForCreditorAgent[2] \*/

**IF** **IsPresentPattern**(MT72, "/PHONBEN/") THEN

{ InstructionForCreditorAgent[2].Code = “PHOB”

**MT72\_To\_MXText**(MT72, "/PHONBEN/", 140; MXText)

**IF** **Length**(MXText) > 0 THEN

InstructionForCreditorAgent[2]. InstructionInformation = MXText

/\* Truncation with “+”sign is included in MT72\_To\_MXText \*/

**ENDIF**

RemainingLength = 140 – LENGTH(MXText)

**IF** **IsPresentPattern**(MT72, "/TELEBEN/") THEN

**IF** RemainingLength < 7 THEN

/\* At least it should be possible to add “/TELB/+” \*/

Flag\_MissingInformation = “true”

EXIT function

**ENDIF**

**MT72\_To\_MXText**(MT72, "/TELEBEN/", 140; MXText)

MXInstructionInformation = Concatenate(“/”, “TELB”, “/”, MXText)

**IF** **Length**(MXInstructionInformation) > RemainingLength THEN

MXInstructionInformation = **Concatenate**(**Substring**(MXInstructionInformation, 1, RemainingLength – 1), “+”)

**ENDIF**

InstructionForCreditorAgent[2]. InstructionInformation = Concatenate (InstructionForCreditorAgent[2]. InstructionInformation, MXInstructionInformation)

**ENDIF** /\* END IF IsPresentPattern(MT72, "/TELEBEN/") \*/

**ELSE** /\* /PHONBEN/ is absent and /TELEBEN/ must be present \*/

InstructionForCreditorAgent[2].Code = “TELB”

**MT72\_To\_MXText**(MT72,"/TELEBEN/", 140; MXText)

InstructionForCreditorAgent[2].InstructionInformation = MXText

}

**ENDIF** /\* IsPresentPattern(MT72, "/PHONBEN/") \*/

**Case** ConfigIndicator = “CodeOnly”

/\* This configuration means that only (/PHONBEN/ or /TELEBEN/) is present with “or” not being exclusive. \*/

/\*It is assumed that /PHONBEN/ and /TELEBEN/ related information is limited to 140 char otherwise a truncation sign “+” is added. In case only one of the codes is present which is probably the case as the code meanings are rather exclusive, and more than 140 char is needed for the related information, the function could be optimised, if need be \*/

**IF** **IsPresentPattern**(MT72, "/PHONBEN/") THEN

InstructionForCreditorAgent[1].Code = “PHOB”

**MT72\_To\_MXText**(MT72,"/PHONBEN/", 140; MXText)

**IF** **Length**(MXText) > 0 THEN

InstructionForCreditorAgent[1]. InstructionInformation = MXText

**ENDIF**

First = “false”

**ENDIF**

**IF** **IsPresentPattern**(MT72, "/TELEBEN/") THEN

**MT72\_To\_MXText**(MT72,"/TELEBEN/", 140; MXText)

**IF** First = “true” Then

InstructionForCreditorAgent[1].Code = “TELB”

**IF** **Length**(MXText) > 0 THEN

InstructionForCreditorAgent[1]. InstructionInformation = MXText

**ENDIF**

**ELSE**  /\* First is not true \*/

InstructionForCreditorAgent[2].Code = “TELB”

**IF** **Length**(MXText) > 0 THEN InstructionForCreditorAgent[2].InstructionInformation = MXText

**ENDIF**

**ENDIF**

**ENDIF**

**Case** ConfigIndicator = “UDLCACC”

/\* /UDLC/ and /ACC/ both present.

/ACC/ is translated to the first occurrence and possibly truncated if length > 140 char.

/UDLC/ is translated to the second occurrence of InstuctionInformation and is expected to use max 1 occurrence otherwise it is truncated.

If /PHONBEN/ or /TELEBEN/ are also present, they are not translated. \*/

MXText = MXACCText

**IF Length** (MXText) > 140 THEN

MXText = **Concatenate**(**Substring**(MXText,1,139),”+”)

**ENDIF**

InstructionForCreditorAgent[1].InstructionInformation = MXText

MXText = **Concatenate**(“/UDLC/”, MXUDLCText)

**IF Length** (MXText) > 140 THEN

MXText = **Concatenate**(**Substring**(MXText,1,139),”+”)

**ENDIF**

InstructionForCreditorAgent[2].InstructionInformation = MXText

**IF** **IsPresentPattern**(MT72, "/TELEBEN/")

Flag\_MissingInformation = “true”

**ENDIF**

**IF** **IsPresentPattern**(MT72, "/PHONBEN/")

Flag\_MissingInformation = “true”

**ENDIF**

**Case** ConfigIndicator = “UDLC”

MXText = Concatenation(“/UDLC/”, MXUDLCText)

**IF Length** (MXText) > 140 THEN

MXText = **Concatenate**(**Substring**(MXText,1,139),”+”)

**ENDIF**

**IF** **IsPresentPattern**(MT72, "/PHONBEN/") OR **IsPresentPattern**(MT72, "/TELEBEN/") THEN

InstructionForCreditorAgent[**2**].InstructionInformation = MXText

**Fill**(**\*\***)(InstructionForCreditorAgent[**1**] )

/\* (**\*\***) Reuse the logic of pseudo- code above from Case ConfigIndicator = “ACCCode” under the section

“Fill in InstructionForCreditorAgent[2]”

BUT **fill** in InstructionForCreditorAgent[**1**] to translate /PHONBEN/ and/or /TELEBEN/. /UDLC/ will always use the last occurrence available. \*/

/\* If /PHONBEN/ and/or /TELEBEN/ is(are) present, the **first occurrence of InstructionInformation** is used. Translation is then similar to the case ConfigIndicator = “ACCCode” but here even if

/PHONBEN/ and /TELEBEN/ are absent, /UDLC/ info is translated to one occurrence of InstructionInformation otherwise the information in the second occurrence is ambiguous and it is not clear if it still part of /UDLC/ or is a textual information resulting from /ACC/ information in MT \*/

**ELSE**

InstructionForCreditorAgent[**1**].InstructionInformation = MXText

**ENDIF**

### 3.3.17 MT\_To\_MXBNF\_TSU\_RI

**Business description**

**Name**

MT\_To\_MXBNF\_TSU\_RI

**Format**

MT\_To\_MXBNF\_TSU\_RI(MT72 ;MXUnstructured RemittanceInformation )

**Input** Field MT72

**Output** MXUnstructuredRemittanceInformation : Max140Text

**Preconditions**

None

**Formal description**

The function extracts the code /BNF/ and /TSU/ from field 72, concatenates the information and copies it to Unstructured remittance information limited to 1 occurrence (max 140 char). If the information cannot be entirely copied, a sign “+” will indicate the truncation. Flag\_MissingInformation is returned if both /BNF/ and /TSU/ are present followed by relevant information and code /TSU/ can only be copied (or code partially copied) without the relevant following information.

/\* Local variables

MXBNFText, MXTSUText, MXTemp1, MXTemp2 : string

Length1, Length2, RemainingLength : number \*/

/\* Extract information following the code \*/

**IF** **IsPresentPattern**(MT72, "/BNF/") THEN

MT72\_To\_MXText(MT72,”/BNF/”, 205; MXBNFText)

**ENDIF**

**IF** **IsPresentPattern**(MT72, "/TSU/") THEN

MT72\_To\_MXText(MT72,”/TSU/”, 205; MXTSUText)

**ENDIF**

MXTemp1 = Concatenation(“/BNF/”, MXBNFText)

MXTemp2 = Concatenation(“/TSU/”, MXTSUText)

RemainingLength = 140

Length1 = **Length**(MXTemp1)

Length2 = **Length**(MXTemp2)

**IF** Length1 > 5 THEN

/\* data present after the code \*/

**IF** Length1 > 140 THEN

MXUnstructuredRemittanceInformation = **Concatenate**(**Substring**(MXTemp1, 1, 139, “+”)

**ELSE**

MXUnstructuredRemittanceInformation = MXTemp1

**ENDIF**

RemainingLength = 140 – **Length**(MXUnstructuredRemittanceInformation)

**ENDIF**

**IF** Length2 > 5 AND RemainingLength > 0 THEN

/\* data present after the code \*/

**IF** RemainingLength < 6 Then

/\* info is not copied because only the code can be copied or partially\*/

Flag\_MissingInformation = “true”

**ELSE**

/\* At least /TSU/ code could be copied plus + 1 char (+) \*/

**IF** Length2 > RemainingLength THEN

MXUnstructuredRemittanceInformation = **Concatenate**(MXUnstructuredRemittanceInformation, **Substring**(MXTemp2, 1, RemainingLength – 1), “+”)

**ELSE**

MXUnstructuredRemittanceInformation = **Concatenate**(MXUnstructuredRemittanceInformation, MXTemp2)

**ENDIF**

**ENDIF**

**ENDIF**

### 3.3.18 MT\_To\_MXField72NewCodeWords

**Business description**

The function analyses Field72 by using the function **ExtractLinesAsIs** (MT72, “/Code/”, “//”) and search for new code words coming from a payment originated in MX like /INTA/, /SVCLVL/, /LOCINS/, /CATPURP/, /PURP/. Extracted information is translated to relevant MX element.

In Field72 there might be up to 2 occurrences of “/INTA/” for IntermediaryAgent2,3.

ServiceLevel /SVCLVL/ can be present up to 3 occurrences. It is expected to have maximum 2 occurrences because the gpi service Identifier code (G00n) is located in MT block 3 and the value “SDVA” is defined in field 23E.

For all other codes, only one occurrence is allowed in MX and therefore only the first one found will be translated.

LocalInstrument (/LOCINS/) is expected to be followed by another value than “CRED”, “CRTS”, “SPAY”, “SPRI”, “SSTD” which are defined in field 23B

CategoryPurpose (/CATPURP/) is expected to be followed by another value than “INTC” or “CORT” which are defined in 23E.

Purpose (/PURP/) can only be present in field72 when coming from a pacs.009.

/\* IMPORTANT for implementors - if the above information can be identified as an ISO code then translation will copy the code to ISO code element. If the ISO code lists are maintained in translator and have not been updated to reflect the current ISO list contents, the missing codes in Translator versus ISO lists will be translated to the Proprietary element until the next maintenance of the lists maintained in Translator \*/

**Name**

MT\_To\_MXField72NewCodeWords

**Format**

**MT\_To\_MXField72NewCodeWords** (MT72 ; MXIntermediaryAgent2, MXIntermediaryAgent3, MXServiceLevel, MXLocalInstrument, MXCategoryPurpose, MXPurpose)

**Input** Field MT72

**Output**

MXIntermediaryAgent2, MXIntermediaryAgent3 typed BranchAndFinancialInstitutionIdentification6.

MXServiceLevel typed ServiceLevel8Choice

MXLocalInstrument typed LocalInstrument2Choice

MXCategoryPurpose typed CategoryPurpose1Choice

MXPurpose typed Purpose2Choice

Where MXIntermediaryAgent2, MXIntermediaryAgent3 correspond to the elements named respectively IntermediaryAgent2 and IntermediaryAgent3 in MX messages.

**Preconditions**

None

**Formal description**

/\* LocalVariables

MTIntermediaryAgent1,MTIntermediaryAgent2, MTServiceLevel, MTLocalInstrument, MTCategoryPurpose, MTPurpose, : string

i : integer

MT72Temp : same structure as Field 72 (6\*35, CRLF, “//” for continuation of previous line) \*/

/\* Search for Intermediary Agent information \*/

MTIntermediaryAgent1 = **ExtractLinesAsIs**(MT72, “/INTA/”, “//”)

/\* Code word is included in the extraction \*/

MT72Temp = **DeletePattern**(MT72, MTIntermediaryAgent1)

MTIntermediaryAgent2 = **ExtractLinesAsIs**(MT72Temp, “/INTA/”, “//”)

/\* Remove the code word /INTA/ \*/

/\* Extracts from MTAgent the information according to the pattern defined in MX\_To\_MT72FullField(2) and fill in the information in the MXAgent \*/

**IF** **Length**(MTIntermediaryAgent1) > 6 THEN

/\* remove code INTA and check if information after \*/

**MT\_To\_MXIntermediaryAgent**(**Substring**(MTIntermediaryAgent1, 7), MXIntermediaryAgent2)

**ENDIF**

**IF** **Length**(MTIntermediaryAgent2) > 6 THEN

/\* remove code INTA and check if information after is present \*/

**MT\_To\_MXIntermediaryAgent**(**Substring**(MTIntermediaryAgent2, 7), MXIntermediaryAgent3)

**ENDIF**

/\* End Intermediary Agent translation \*/

/\* Search for ServiceLevel information up to 3 occurrences but max 2 expected (gpi codes handled elsewhere) and copy to a new occurrence of MXServiceLevel \*/

MT72Temp = MT72

**For i = 1 to 3**

MTServiceLevel = **ExtractLinesAsIs**(MT72Temp, “/SVCLVL/”, “//”)

/\* code word is included in the extraction \*/

/\* prepare the string for the next search of code by removing the info extracted above included the code word\*/

MT72Temp = **DeletePattern**(MT72Temp, MTServiceLevel)

/\*Remove code word \*/

MTServiceLevel = **Substring**(MTServiceLevel, 9)

**IF** **Length**(MTServiceLevel) > 0 AND MTServiceLevel <> “SDVA” AND MTServiceLevel **HasPattern** different “G00n” (n is an integer) THEN

**{ IF** **NumberOfFreeOccurrence**(MXServiceLevel) = 0 THEN

/\* no more occurence available to copy information. Max 3 occurences \*/

Flag\_MissingInformation = “true”

**ELSE**

/\* Create a new MXServiceLevel Occurrence – let’s call it “n” \*/

**IF** **WithinList(**MTServiceLevel, ISOServiceLevelCode\*) THEN

MXServiceLevel[n].Code = MTServiceLevel

**ELSE**

**IF LENGTH**(MTServiceLevel) > 35 THEN

MXServiceLevel[n].Proprietary = **Concatenate**(**Substring**(MTServiceLevel, 1, 34), “+”)

**ELSE**

MXServiceLevel[n].Proprietary = MTServiceLevel

**ENDIF**

/\* only the 35 first characters are copied. As coming from MX, it is not expected to have longer string but just to avoid error in translation, length is limited to 35 \*/

**ENDIF**

**ENDIF** /\* NumberOfFreeOccurrence(MXServiceLevel) = 0 \*/

**}**

\*ISOServiceLevelCode is defined on [ISO 20022 Site](https://www.iso20022.org/external_code_list.page), [External Code Sets spreadsheet](https://www.iso20022.org/sites/default/files/documents/External_code_lists/ExternalCodeSets_1Q2019_May2019_v1.xls), sheet 15-ServiceLevel

**ENDIF** /\* IF LENGTH(MTServiceLevel) > 0 \*/

**Next i**

/\* End Service level code \*/

/\* Search for Local Istrument information other than {CRED, CRTS, SPAY, SPRI, SSTD} – only 1 occurrence is allowed\*/

MTLocalInstrument = **ExtractLinesAsIs**(MT72, “/LOCINS/”, “//”)

/\* code word is included in the extraction \*/

/\*Remove code word \*/

MTLocalInstrument = **Substring**(MTLocalInstrument, 9)

**IF** **Length**(MTLocalInstrument) > 0 AND MTLocalInstrument **NOT IsInList** {CRED,CRTS,SPAY, SPRI, SSTD} THEN

**{ IF** **NumberOfFreeOccurrence**(MXLocalInstrument)= 0 THEN

/\* no more occurence available to copy information. Max 1 occurence \*/

Flag\_MissingInformation = “true”

**ELSE**

/\* Create a new MXLocalInstrument Occurrence \*/

**IF** **WithinList**(MTLocalInstrument, ISOLocalInstrumentCode\*) THEN

MXLocalInstrument [1].Code = MTLocalInstrument

**ELSE**

**IF Length**(MTLocalInstrument) > 35 THEN

MXLocalInstrument [1].Proprietary = **Concatenate**(**Substring**(MTLocalInstrument,1,34), “+”)

**ELSE**

MXLocalInstrument [1].Proprietary = MTLocalInstrument

/\* only the 35 first characters are copied. As coming from MX, it is not expected to have longer string but just to avoid error in translation, length is limited to 35 \*/

**ENDIF**

**ENDIF**

**ENDIF** /\*NumberOfFreeOccurrence(MXLocalInstrument) = 0 \*/

**}**

\*ISOLocalInstrumentCode is defined on [ISO 20022 Site](https://www.iso20022.org/external_code_list.page), [External Code Sets spreadsheet](https://www.iso20022.org/sites/default/files/documents/External_code_lists/ExternalCodeSets_1Q2019_May2019_v1.xls), sheet 7-LocalInstrument

**ENDIF** /\* IF LENGTH(MTLocalInstrument) > 0 \*/

/\* End LocalInstrument code \*/

/\* Search for Category Purpose information other than {INTC, CORT} – only 1 occurrence is allowed\*/

MTCategoryPurpose = **ExtractLinesAsIs**(MT72, “/CATPURP/”, “//”)

/\* code word is included in the extraction \*/

/\*Remove code word \*/

MTCategoryPurpose = **Substring**(MTCategoryPurpose, 10)

**IF** **Length**(MTCategoryPurpose) > 0 AND MTCategoryPurpose **NOT IsInList** {INTC,CORT} THEN

**{ IF** **NumberOfFreeOccurrence**(MXCategoryPurpose) = 0 THEN

/\* no more occurence available to copy information. Max 1 occurence \*/

Flag\_MissingInformation = “true”

**ELSE**

/\* Create a new MXCategoryPurpose Occurrence \*/

**IF** **WithinList**(MTCategoryPurpose, ISOCategoryPurposeCode\*) THEN

MXCategoryPurpose [1].Code = MTCategoryPurpose

**ELSE**

**IF Length**(MTCategoryPurpose) > 35 THEN

MXCategoryPurpose [1].Proprietary = **Concatenate**(**Substring**(MTCategoryPurpose, 1, 34), “+”)

**ELSE**

MXCategoryPurpose [1].Proprietary = MTCategoryPurpose

**ENDIF**

/\* only the 35 first characters are copied. As coming from MX, it is not expected to have longer string but just to avoid error in translation, length is limited to 35 \*/

**ENDIF**

**ENDIF** /\*NumberOfFreeOccurrence(MXCategoryPurpose) = 0 \*/

**}**

\*ISOCategoryPurposeCode is defined on [ISO 20022 Site](https://www.iso20022.org/external_code_list.page), [External Code Sets spreadsheet](https://www.iso20022.org/sites/default/files/documents/External_code_lists/ExternalCodeSets_1Q2019_May2019_v1.xls), sheet 4-CategoryPurpose

**ENDIF** /\* IF LENGTH(MTCategoryPurpose) > 0 \*/

/\* End CategoryPurpose code \*/

/\* Search for Purpose information – only 1 occurrence is allowed \*/

MTPurpose = **ExtractLinesAsIs**(MT72, “/PURP/”, “//”)

/\* code word is included in the extraction \*/

/\*Remove code word \*/

MTPurpose = **Substring**(MTCategoryPurpose, 7)

**IF** **Length**(MTPurpose) > 0 THEN

**{ IF** **NumberOfFreeOccurrence**(MXPurpose)= 0 Then

/\* no more occurence available to copy information. Max 1 occurence \*/

Flag\_MissingInformation = “true”

**ELSE**

/\* Create a new MXPurpose Occurrence \*/

**IF** **WithinList**(MTPurpose, ISOPurposeCode\*) Then

MXPurpose [1].Code = MTPurpose

**ELSE**

**IF Length**(MTPurpose) > 35 THEN

MXPurpose [1].Proprietary = **Concatenate**(**Substring**(MTPurpose, 1, 34), ”+”)

**ELSE**

MXPurpose [1].Proprietary = MTPurpose

**ENDIF**

/\* only the 35 first characters are copied. As coming from MX, it is not expected to have longer string but just to avoid error in translation, length is limited to 35 \*/

**ENDIF**

**ENDIF** /\* NumberOfFreeOccurrence(MXPurpose) = 0 \*/

**}**

\*ISOCategoryPurposeCode is defined on [ISO 20022 Site](https://www.iso20022.org/external_code_list.page), [External Code Sets spreadsheet](https://www.iso20022.org/sites/default/files/documents/External_code_lists/ExternalCodeSets_1Q2019_May2019_v1.xls), sheet 11-Purpose

**ENDIF** /\* IF LENGTH(MTPurpose) > 0 \*/

/\* End Purpose code \*/

### 3.3.19 MT\_To\_MX\_Serial\_CoverScenario

**Business description**

The function analyses the fields 53a and 54a in order to identify if the scenario is a cover or serial payment.

**Name**

MT\_To\_MX\_Serial\_CoverScenario

**Format**

**MT\_To\_MX\_Serial\_CoverScenario** (53a, 54a, SenderBIC, ReceiverBIC;MTScenario)

**Input**

53a, 54a fields from MT message

SenderBIC and ReceiverBIC : BIC (11) (Identifier Code)

**Output**

MTScenario : value {SERIAL, COVER}

**Preconditions**

None

**Formal description**

Based on different combinations of field 53a and 54a, the table below aims at identifying the scenario “serial” versus “cover”. The code “SERIAL” or “COVER” is returned.



### 3.3.20 MT\_To\_MXReturn72

**Business description**

The function translates the following lines from Field 72 in RETN payment :

-Line 2 /2!c2!n/[29x] with the return reason code (mandatory)

-Line 6 /TEXT/ with additional information (optional)

The MT reason code will be converted to the MX Reason code equivalent. If there is no MX Reason Code equivalent or if the MT reason code is a bilaterally agreed code (format X1!c2!n) then the code NARR is used in MX and the MT reason code is copied to Additional Information. If there is narrative free text (from Line 2 or line 6), the MT reason code is separated from the narrative information with “/” as following “MTCode[/Textual]”

If the original message used for the Return is pacs.004, then the Line 2 can also have the following structure (refer to MX\_To\_MT72RETN):

/XT99/MXReasonCode/[Text1]

The table to be used to convert the reason code is described in excel pacs.002 to MT199 MT299 REJT in sheet “Error Codes RETN REJT”.

**Name**

MT\_To\_MXReturn72

**Format**

**MT\_To\_MXReturn72** (Field 72; ReturnReasonInformation)

**Input**

Field 72

**Output**

ReturnReasonInformation typed PaymentReturnReason6

**Preconditions**

Field 72 with Line 1 : /RETN/2!n

**Formal description**

/\* Extract narrative information from Line 2 and Line 6

The numbers refer to the description of Field 72 in UHB but as some lines are optional (Line 1,2,3 are mandatory, Line 4,5,6 are optional) , it does not refer necessarily to the line number in a specific instance of Field 72 in a message \*/

/\* Local variables

AdditionalInformation, Line2, Line6 : string

MTReason, MXReason, MXReasonCode, MXStructure : string

LineNumber, MaxLength :integer

MXEquivalent : Boolean \*/

/\* Extract from Line 2, Reason code and the narrative information. Remove *CRLF.*

MT pattern “/2!c2!n/” is equivalent to the regular expression “/[0-9A-Z]{2}[0-9]{2}/” \*/

Line2 = Field72.Line[2]

/\* Search for structure /XT99/MXReasonCode/Text1 or /2!c2!n/ \*/

**IF** **IsPresentPattern**(Line2, “/XT99/”) THEN

MXStructure = **ExtractFromPattern**(Line2,”/XT99/”)

MXReasonCode = **ExtractTillPattern**(MXStructure,”/”)

AdditionalInformation = **ExtractFromPattern**(MXStructure,”/”)

MaxLength = 35 – 6 – **Length**(MXReasonCode)-1

/\* MaxLength is the max length of AdditionalInformation ie Text1 \*/

**ELSE**

AdditionalInformation = **ExtractFromPattern**(Line2,”/”[0-9A-Z]{2}[0-9]{2}”/”)

MaxLength = 29

**ENDIF**

/\* Search for narrative text following codeword “/TEXT/”, position depends on Field 72 instance. Multiple lines separated by “//” is allowed \*/

Line6 = **ExtractLines**(Field72, “/TEXT/”,”//”)

**IF Length**(Line6) > 0 THEN

/\* Remove the “/TEXT/” \*/

Line6 = **ExtractFromPattern**(Line6,”/TEXT/”)

**ENDIF**

/\* Check if information is present after “/TEXT/” and concatenate the narrative information \*/

**IF Length**(Line6)> 0 THEN

**IF Length**(AdditionalInformation) > 0 THEN

**IF Length**(AdditionalInformation)< MaxLength THEN

/\* add a space between the 2 narrative texts \*/

AdditionalInformation = **Concatenate**(AdditionalInformation, SPACE, Line6)

**ELSE**

AdditionalInformation = **Concatenate**(AdditionalInformation, Line6)

**ENDIF**

**ELSE**

AdditionalInformation = Line6

**ENDIF**

**ENDIF**

**IF Length**(MXReasonCode) > 0 THEN

/\* case /XT99/ from pacs.004 \*/

**IF Length**(MXReasonCode)= 4 **AND** **WithinList**(MXReasonCode, ExternalReturnReason1Code) THEN

/\* ISO list defined on ISO 20022 site,ExternalCodeSets, Sheet 13 \*/

ReturnReasonInformation.Reason.Code = MXReasonCode

**ELSE**

/\* This case should not happen but foreseen to smooth translation \*/

ReturnReasonInformation.Reason.Code = “NARR”

**IF Length**(AdditionalInformation) > 0 THEN

AdditionalInformation = **Concatenate**(MXReasonCode,”/”, AdditionalInformation)

**ELSE**

AdditionalInformation = MXReasonCode

**ENDIF**

**ENDIF**

**ELSE**

**{** /\* case /2!c2!n/ \*/

MTReason = **ExtractPattern**(Line2,”/”[0-9A-Z]{2}[0-9]{2}”/”)

MTReason = **TrimLeft**(MTReason, “/”)

MTReason = **TrimRight**(MTReason, “/”)

MXReason = “NARR”

MXEquivalent = “false”

**IF IsPresent**(MTReason, Table Col A) THEN

/\* extract the line number in the table \*/

LineNumber = **GetLineNumber**(MTReason)

/\*Read the corresponding value in col C \*/

**IF IsMXErrorCodePresent** (Table, LineNumber, Col C)THEN

MXReason = **GetValue**(LineNumber, Col H)

MXEquivalent = “True”

**ENDIF**

**ENDIF**

/\* If MTReason has no equivalent or not present in the list, it is concatenated to Additional Information \*/

**IF** MXEquivalent= ”false” THEN

**IF Length**(AdditionalInformation)> 0 THEN

AdditionalInformation= **Concatenate**(MTReason,”/”,AdditionalInformation)

**ELSE**

AdditionalInformation = MTReason

**ENDIF**

**ENDIF**

/\* fill in MX Return Reason Information \*/

ReturnReasonInformation.Reason.Code = MXReason

**}** /\* End of ELSE case handling /2!c2!n/ \*/

**/\* For all cases, copy Additional Information \*/**

**IF Length**(AdditionalInformation)> 0 THEN

**IF Length**(AdditionalInformation) > 210 THEN

AdditionalInformation = **Concatenate**(**Substring**(AdditionalInformation,1,209),+)

**ENDIF**

**IF Length**(AdditionalInformation)> 105 THEN

ReturnReasonInformation.AdditionalInformation[1]= **Substring**(AdditionalInformation, 1,105)

ReturnReasonInformation.AdditionalInformation[2]= **Substring**(AdditionalInformation, 106)

**ELSE**

ReturnReasonInformation.AdditionalInformation[1]= AdditionalInformation

**ENDIF**

**ENDIF**

### 3.3.21 MT\_To\_ MXRegulatoryReporting2

**Name**

MT\_To\_MXRegulatoryReporting2

**Business description**

The function searches for the code word /BENEFRES/ and /ORDERRES/ and extracts the CountryOfResidence. If it is a valid country code, it is translated to the Creditor (or Debtor)/CountryOfResidence.

The function is a subset of MT\_To\_MXRegulatoryReporting.

Field 77B has the format : Narrative (3\*35x) OR

Line 1 : /8a/2!a[//AdditionalInformation]

Line 2-3 : [//continuation of additional information]

Where /8a/ is one of the code /BENEFRES/ or /ORDERRES/

2!a is the country code of the Residence of the Beneficiary/Creditor or Debtor

**Format**

**MT\_To\_MXRegulatoryReporting2**(MTRegulatoryReporting ; MXDebtor, MXCreditor)

**Input**

MTRegulatoryReporting: regulatory reporting details in an MT field 77B (format 3\*35x).

**Output**

MXDebtor, MXCreditor are typed PartyIdentification135

**Preconditions**

None.

**Formal description**

/\*MTRegulatoryReporting is defined by 3 lines of 35 characters.

MTRegulatoryReporting[1] indicates the first line not including the Carriage Return Line Feed “*CRLF*” needed as separator

between consecutive lines in an MT field with multiple lines format\*/

/\* Local variables

CreditorLine, DebtorLine, CountryCode : string

CreditorResPattern, DebtorResPattern : pattern \*/

CreditorResPattern = “/BENEFRES/” followed by exactly 2 alphabetic char, upper case (which must be an ISO Country code)

DebtorResPattern = “/ORDERRES/” followed by exactly 2 alphabetic char, upper case (which must be an ISO Country code)

/\* The 2 codes can be present as per customer samples \*/

CreditorLine = **ExtractPattern**(MTRegulatoryReporting, CreditorResPattern)

DebtorLine = **ExtractPattern**(MTRegulatoryReporting, DebtorResPattern)

/\* Fill in the Party Country Of Residence \*/

**IF** **Length**(CreditorLine) > 0 THEN

CountryCode = Substring(CreditorLine,11,2)

**IF** **IsValidCountryCode**(CountryCode) THEN

/\* Meaning valid ISO country code \*/

MXCreditor.CountryOfResidence = CountryCode

**ENDIF**

**ENDIF**

**IF** **Length**(DebtorLine) > 0 THEN

CountryCode = Substring(DebtorLine,11,2)

**IF** **IsValidCountryCode**(CountryCode) THEN

/\* Meaning valid ISO country code \*/

MXDebtor.CountryOfResidence = CountryCode

**ENDIF**

**ENDIF**

### 3.3.22 MT\_To\_MXReject72

**Business description**

The function translates the following lines from Field 72 in REJT payment :

-Line 2 /2!c2!n/ with the reject reason (code or proprietary /X1!c2!n/) (mandatory)

-Line 6 with additional information (optional)

The MT reason code will be converted to the MX Reason code equivalent. If there is no MX Reason Code equivalent or if the MT reason code is a bilaterally agreed code (format X1!c2!n) then the MT Reason code is copied to MX Reason/Proprietary.

If the original message used for the Reject is pacs.002, then the Line 2 can also have the following structure (refer to MX\_To\_MT72REJT):

/XT99/MXReasonCode/Text1 or /XT99/MXReasonProprietary/Text1

The table to be used to convert the reason code is described in excel pacs.002 to MT199 MT299 REJT in sheet “Error Codes RETN REJT”.

**Name**

MT\_To\_MXReject72

**Format**

**MT\_To\_MXReturn72** (Field 72; StatusReasonInformation)

**Input**

Field 72

**Output**

StatusReasonInformation typed StatusReasonInformation12

**Preconditions**

Field 72 with Line 1 : /REJT/2!n

**Formal description**

/\* Extract reason code and narrative information from Line 2 and Line 6

The numbers refer to the description of Field 72 in UHB but as some lines are optional (Line 1,2,3 are mandatory, Line 4,5,6 are optional) , it does not refer necessarily to the line number in a specific instance of Field 72 in a message \*/

/\* Local variables

AdditionalInformation, Line2, Line6 : string

MTReason, MXReason, MXReasonCodeOrProprietary, MXStructure : string

LineNumber, MaxLength :integer

MXEquivalent : Boolean \*/

/\* Extract from Line 2, Reason code and the narrative information. Remove *CRLF.*

MT pattern “/2!c2!n/” is equivalent to the regular expression “/[0-9A-Z]{2}[0-9]{2}/” \*/

Line2 = Field72.Line[2]

/\* Search for structure /XT99/MXReasonCode/Text1 or /XT99/MXReasonProprietary/Text1 or /2!c2!n/ \*/

**IF** **IsPresentPattern**(Line2, “/XT99/”) THEN

MXStructure = **ExtractFromPattern**(Line2,”/XT99/”)

MXReasonCodeOrProprietary = **ExtractTillPattern**(MXStructure,”/”)

AdditionalInformation = **ExtractFromPattern**(MXStructure,”/”)

/\* Assumption is that there is no “/” within MXReasonProprietary \*/

MaxLength = 35 – 6 – **Length**(MXReasonCodeOrProprietary)-1

/\* MaxLength is the max length of AdditionalInformation ie Text1 \*/

**ELSE**

AdditionalInformation = **ExtractFromPattern**(Line2,/[0-9A-Z]{2}[0-9]{2}/)

MaxLength = 29

**ENDIF**

/\* Search for narrative text following codeword “/TEXT/”, position depends on Field 72 instance. Multiple lines separated by “//” is allowed \*/

Line6 = **ExtractLines**(Field72, “/TEXT/”,”//”)

/\* Remove the “/TEXT/” \*/

**IF Length**(Line6) > 0 THEN

Line6 = **ExtractFromPattern**(Line6,”/TEXT/”)

**ENDIF**

/\* concatenate the narrative information \*/

**IF Length**(Line6)> 0 THEN

**IF Length**(AdditionalInformation) > 0 THEN

**IF Length**(AdditionalInformation)< MaxLength THEN

/\* add a space between the 2 narrative texts \*/

AdditionalInformation = **Concatenate**(AdditionalInformation, SPACE, Line6)

**ELSE**

AdditionalInformation = **Concatenate**(AdditionalInformation, Line6)

**ENDIF**

**ELSE**

AdditionalInformation = Line6

**ENDIF**

**ENDIF**

/\* Fill in MX Additional Information \*/

**IF Length**(AdditionalInformation)> 0 THEN

**IF Length**(AdditionalInformation) > 210 THEN

AdditionalInformation = **Concatenate**(**Substring**(AdditionalInformation,1,209),+)

**ENDIF**

**IF Length**(AdditionalInformation)> 105 THEN

StatusReasonInformation.AdditionalInformation[1]= **Substring**(AdditionalInformation, 1,105)

StatusReasonInformation.AdditionalInformation[2]= **Substring**(AdditionalInformation, 106)

**ELSE**

StatusReasonInformation.AdditionalInformation[1]= AdditionalInformation

**ENDIF**

**ENDIF** /\* End length(AdditionalInformation) \*/

/\* Copy the Reason code \*/

**IF Length**(MXReasonCodeOrProprietary) > 0 THEN

/\* case /XT99/ from pacs.002 \*/

**IF Length**(MXReasonCodeOrProprietary)= 4 **AND** **WithinList**(MXReasonCodeOrProprietary, ExternalStatusReaon1Code) THEN

/\* ISO list defined on ISO 20022 site,ExternalCodeSets, Sheet 16 \*/

StatusReasonInformation.Reason.Code = MXReasonCodeOrProprietary

**ELSE**

StatusReasonInformation.Reason.Proprietary = MXReasonCodeOrProprietary

**ENDIF**

**ELSE**

/\* Structure like /2!c2!n/ \*/

/\* check if the MT Reason in line2 and defined in MT UHB list has a MX equivalent.The table to be used to convert the reason code is referred above.

If yes, then it is converted to MX Reason and copied to Reason/Code else it is copied to Reason/Proprietary \*/

MTReason = **ExtractPattern**(Line2,/[0-9A-Z]{2}[0-9]{2}/)

MTReason = **TrimLeft**(MTReason, “/”)

MTReason = **TrimRight**(MTReason, “/”)

MXEquivalent = “false”

**IF IsPresent**(MTReason, Table Col A) THEN

/\* extract the line number in the table \*/

LineNumber = **GetLineNumber**(MTReason)

/\*Read the corresponding value in col C \*/

**IF IsMXErrorCodePresent** (Table, LineNumber, Col C)THEN

MXReason = **GetValue**(LineNumber, Col H)

MXEquivalent = “True”

**ENDIF**

**ENDIF**

**IF** MXEquivalent= ”false” THEN

StatusReasonInformation.Reason.Proprietary = MTReason

**ELSE**

StatusReasonInformation.Reason.Code = MXReason

**ENDIF**

**ENDIF /\* EndIF Length**(MXReasonCodeOrProprietary) > 0 \*/

### 3.3.23 MT\_To\_MXPurpose

**Name**

MT\_To\_MXPurpose

**Business description**

The function extracts information from field 70 with code word /PURP/ (eg., in MT103). This scenario can only occur if MT103 results from a previous translation pacs.008 to MT103 where MX Purpose (Code or Propriatery) was translated to field 70.

**Format**

**MT\_To\_MXPurpose**(MTField70, MXLocation)

**Input**

MTField70 : Field70 in MT103

**Output**

MXLocation is MX CreditTransferTransactionInformation typed *CreditTransferTransaction39*

**Preconditions** None.

**Formal description**

/\* Local variables

PurposeString : string

PurposePattern, ToDeletePattern : string

PatternTable[] : table of string

i : integer

\*/

PurposePattern = “/PURP/”

ToDeletePattern = “/”

/\* Search for the codewords /PURP/ and extract information following the codeword.

Possible other codewords present in field 70 : /ULTB/,:ULTD/, /ROC/, /URI/, /RELID/, /SRI/. \*/

PatternTable[1] = “/ULTB/”

PatternTable[2] = “/ULTD/”

PatternTable[3] = “/PURP/”

PatternTable[4] = “/ROC/”

PatternTable[5] = “/URI/”

PatternTable[6] = “/RELID/”

PatternTable[7] = “/SRI/”

PurposeString= **ExtractbetweenPattern**(MTField70,PurposePattern,

{PatternTable[1], PatternTable[2], PatternTable[3], PatternTable[4], PatternTable[5], PatternTable[6], PatternTable[7]})

/\* The additional “//” are removed if they occur at the end of the extracted information. SubfunctionTrimRight is defined in MT\_To\_MXUltimateParty \*/

Call **SubfunctionTrimRight**(PurposeString, PurposePattern, MTField70,ToDeletePattern, PatternTable[]; PurposeString)

**IF Length**(PurposeString) > 0

/\* check if it is ISO Code, see reference below \*/

**IF WithinList**(PurposeString, ISOPurposeCode\*)THEN

MXLocation.Pupose.Code = PurposeString

**ELSE**

**IF Length**(PurposeString) > 35 THEN

MXLocation.Pupose.Proprietary = **Concatenate**(**Substring**(PurposeString,1,34), “+”)

**ELSE**

MXLocation.Pupose.Proprietary = PurposeString

/\* only the 35 first characters are copied. As coming from MX, it is not expected to have longer string but just to avoid error in translation, length is limited to 35 \*/

**ENDIF**

**ENDIF**

**ENDIF**

\*ISOPurposeCode is defined on [ISO 20022 Site](https://www.iso20022.org/external_code_list.page), [External Code Sets spreadsheet](https://www.iso20022.org/sites/default/files/documents/External_code_lists/ExternalCodeSets_1Q2019_May2019_v1.xls), sheet 11-Purpose \*

### 3.3.24 MT\_To\_MXRINewCodeWords

**Name**

MT\_To\_MXRINewCodeWords

**Business description**

The function extracts information from field 70 with code word /URI/, /RELID/ or /SRI/ . This scenario can only occur from a previous translation (eg., pacs.008 to MT103) where MX Unstructured Remittance Information or Related Remittance Information was translated to field 70. If MX structured RI was present in MX then it is indicated by the presence of /SRI/+ as the Structured RI is not translated. The 3 code words are exclusive meaning they cannot occur all together in field 70.

**Format**

**MT\_To\_MXRINewCodeWords** (MTField70; MXRelatedRemittanceInformation, MXRemittanceInformation)

**Input**

MTField70 : Field70 in MT103

**Output**

MXRelatedRemittanceInformation typed RemittanceLocation7

MXRemittanceInformation typed RemittanceInformation16

**Preconditions** None.

**Formal description**

/\* Local variables

URemittanceString, SRemittanceString, RelatedIDString : string

URIPattern, SRIPattern, RelatedRIPattern, ToDeletePattern : string

PatternTable[] : table of string

i : integer

\*/

URIPattern = “/URI/”

SRIPattern = “/SRI/+”

RelatedRIPattern = “/RELID/”

ToDeletePattern = “/”

/\* Search for the codewords listed above and extract information following the codewords.The search is generic but the codewords above should be the last ones in the field 70.

Possible other codewords present in field 70 : /ULTB/,/ULTD/, /PURP/. Note that the code /ROC/ if present in Field 70 is translated to EndToEndID. If it occurs after /URI/, it will be translated to Unstructured RI (as might be part of that information as other codewords like “INV” for example \*/

PatternTable[1] = “/ULTB/”

PatternTable[2] = “/ULTD/”

PatternTable[3] = “/PURP/”

PatternTable[4] = “/URI/”

PatternTable[5] = “/RELID/”

PatternTable[6] = “/SRI/+”

URemittanceString= **ExtractBetweenPattern**(MTField70,URIPattern,

{PatternTable[1], PatternTable[2], PatternTable[3], PatternTable[4], PatternTable[5], PatternTable[6]})

/\* The additional “//” are removed if they occur at the end of the extracted information except for RI information which is expected to be the last one as per construction in MX to MT. In order to cater for possible changes in the priority order in MX to MT translation, the check is still done through the function SubfunctionTrimRight defined in MT\_TO\_MXultimateParty. However it is important to keep /ROC/ before URI if it is not originally part of the MT RI. \*/

Call **SubfunctionTrimRight**(URemittanceString, URIPattern, MTField70,ToDeletePattern, PatternTable[]; URemittanceString)

RelatedIDString = **ExtractBetweenPattern**(MTField70,RelatedRIPattern,

{PatternTable[1], PatternTable[2], PatternTable[3], PatternTable[4], PatternTable[5], PatternTable[6]})

Call **SubfunctionTrimRight**(RelatedIDString, RelatedRIPattern, MTField70,ToDeletePattern, PatternTable[]; RelatedIDString)

**IF IsPresentPattern**(MTField70, SRIPattern) THEN

/\* search in field 70 spread on Multilines if SRIPattern is present \*/

SRemittanceString = SRIPattern

**ENDIF**

/\* It is expected that only one of the string is filled, possibly none of them. If both are filled, then only one can be translated as their corresponding data in MX are exclusive. In that case the flag Missing information will get the value “True” \*/

**IF Length**(URemittanceString) > 0 THEN

MxRemittanceInformation.Unstructured = URemittanceString

**IF Length**(SRemittanceString) > 0 THEN

Flag\_MissingInformation = “True”

**ENDIF**

**IF Length**(RelatedIDString) > 0 THEN

Flag\_MissingInformation = “True”

**ENDIF**

**ELSEIF Length**(SRemittanceString) > 0 THEN

MXRemittanceInformation.Unstructured = SRemittanceString

**IF Length**(RelatedIDString) > 0 THEN

Flag\_MissingInformation = “True”

**ENDIF**

**ELSEIF** **Length**(RelatedIDString) > 0 THEN

**IF Length**(RelatedIDString) > 35 THEN

/\* This should not occur \*/

RelatedIDString = **Concatenate**(**Substring**(RelatedIDString,1,34),”+”)

**ENDIF**

MXRelatedRemittanceInformation.RemittanceIdentification= RelatedIDString

**ENDIF**

### 3.3.25 MT\_To\_MXField79

**Name**

MT\_To\_MXField79

**Business description**

The function extracts information from field 79 in MT192/MT292 and translates to camt.056 Cancellation Reason Information. From a previous translation camt.056 to MT, the code /UETR/ follows by a valid UETR number might be present.

The format of field 79 is either

-narrative 35\*50x

or

-may be structured as follows

Line 1: /4!c/[additional information] where 4!c is the cancellation reason code

Lines 2-35: [//continuation of additional information

[Last Line /UETR/UETRValue]

Additional information can be truncated. This will be indicated with “+” as last character.

Note that if the continuation sign “//” is not used while /4!c/ is used on the first line, this will result in a truncation of the next lines as it is not expected in the current translation to have a mix of both formats. If needed and if a CR is requested by customer, this could be improved and aligned with MT\_To\_MXField76.

**Format**

**MT\_To\_MXField79** (MTField79, MTUETR ; MXOriginalUETR, MXCancellationReasonInformation)

**Input**

MTField79 : Field79 in MT192/MT292

MTUETR: Block3/EndToEndReference/UniqueEndToEndTransactionReference

**Output**

MXOriginalUETR typed UUIDv4Identifier

MXCancellationReasonInformation typed PaymentCancellationReason5

**Preconditions** None.

**Formal description**

/\* Local variables

ListOfCode : list of string 4 char

MTCode, MXCode, MXAdditionalInformation : string \*/

ListOfCode = {AGNT, AM09, COVR, CURR, CUST, CUTA, DUPL, FRAD, TECH, UPAY,NARR}

/\*Search for UETR information \*/

**IF** **IsPresentPattern**(MTField79, “/UETR/”[a-f0-9]{8}-[a-f0-9]{4}-4[a-f0-9]{3}-[89ab][a-f0-9]{3}-[a-f0-9]{12})THEN

**IF Length**(MTUETR)= 0 THEN

/\* Meaning no UETR in the MT Header. IF UETR is present in MT Header, it is extracted from there. \*/

MXOriginalUETR = **ExtractPattern**(MTField79,[a-f0-9]{8}-[a-f0-9]{4}-4[a-f0-9]{3}-[89ab][a-f0-9]{3}-[a-f0-9]{12})

**ELSE**

/\* Value filled from elsewhere but reinforced here to avoid to overwrite MXOriginalUETR with empty string \*/

MXOriginalUETR = MTUETR

**ENDIF**

MTField79 **= DeletePattern(**MTField79, “/UETR/”[a-f0-9]{8}-[a-f0-9]{4}-4[a-f0-9]{3}-[89ab][a-f0-9]{3}-[a-f0-9]{12})

/\* Remove the line with UETR including the CRLF, if present \*/

**ENDIF**

/\* If “/UETR/”UETRValue is not present in field 79 and is present in the MT Header, it is translated from another function in the excel file. If it is present at both places, then UETR from Header is translated and also filled in this function to avoid empty output parameter \*/

**IF IsEmpty**(MTField79) THEN

MXCancellationReasonInformation.Reason.Code = “NARR”

MXCancellationReasonInformation.AdditionalInformation = “NOTPROVIDED”

**ELSEIF Substring**(MTField79 Line[1], 1,1) = “/” AND

**Substring**(MTField79 Line[1], 6,1) = “/” AND

**WithinList(Substring**(MTField79 Line[1], 2,4), ListOfCode)

/\* the first line of Field 79 is screened \*/

MTCode = **Substring**(MTField79 Line [1],1,6)

/\*extract the code including slash \*/

**Call SubfunctionCode79**

/\* function defined below \*/

**ELSE**

**Call SubfunctionNarrative79**

/\* function defined below \*/

**ENDIF**

/\* Subfunctions are described below \*/

**SubfunctionCode79**

MXAdditionalInformation = **ExtractLines**(MTField79,MTCode, “//)

/\*Remove the MTCode \*/

MXAdditionalInformation= **Substring**(MXAdditionalInformation, 7)

MXCode = **Substring**(MTCode,2,4)

/\* Remove the unexpected spaces added by replacing “//” by a space in the lines concatenation, for example if a line contains just “//” or if after the code on the first line there is no other information on that line and line 2 is not empty\*/

MXAdditionalInformation = **TrimRight**(MXAdditionalInformation,Space)

MXAdditionalInformation = **TrimLeft**(MXAdditionalInformation,Space)

**IF length**(MXAdditionalInformation) > 210 THEN

MXAdditionalInformation = **Concatenate**(**Substring**(MXAdditionalInformation, 1,209), “+”)

**ENDIF**

/\* Copy cancellation reason to MX \*/

MXCancellationReasonInformation.Reason.Code = MXCode

**IF Length**(MXAdditionalInformation) > 105 THEN

MXCancellationReasonInformation.AdditionalInformation[1] = **Substring**(MXAdditionalInformation,1,105)

MXCancellationReasonInformation.AdditionalInformation[2] = **Substring**(MXAdditionalInformation,106)

**ELSEIF Length(**MXAdditionalInformation) > 0 THEN

MXCancellationReasonInformation.AdditionalInformation[1]= MXAdditionalInformation

**ELSE**

/\* Length(MXAdditionalInformation) = 0 \*/

**IF** MXCode = “NARR” THEN

MXCancellationReasonInformation.AdditionalInformation[1]= “NOTPROVIDED”

/\* With “NARR”, AdditionalInformation is mandatory in MX \*/

**ENDIF**

**ENDIF**

/\*End **SubfunctionCode79** \*/

**SubfunctionNarrative79**

/\* The translation will just concatenate the lines from Field 79. If the length of a line is less than 50 char, concatenation with the next line will add a space between the 2 lines otherwise the 2 lines are just concatenated without space as it is considered that the next line is the continuation of the previous line. If "//" is used at the beginning of a line (line 2 to 35), it will be deleted as it is interpretated as the continuation of the previous line, although the "//" could be used as normal characters as well (but small risk to be also at the beginning of the line). Note that “//” being used as continuation characters in a free format means that 2 formats are mixed up) \*/

/\* Local variable

MXString, PreviousLine, Line : string

index : integer

\*/

/\* Assumption is that there is at minimum 1 line otherwise the translation subfunction is not called.

For each line before the next concatenation, add a space at the end of the line if less than 50 char are used (length check done before removing “//”). \*/

/\* Extract MT information\*/

MXString = **TrimRight**(MTField79.Line[1], CRLF)

/\* Remove CRLF if present \*/

**For** index = 2 to **NumberOfOccurrences**(MTField79)

/\* NumberOfoccurences being number of lines of MTField79 after having possibly removed /UETR/ line as explained above\*/

Line = **TrimRight(**MTField79.Line[index],CRLF)

PreviousLine = **TrimRight(**MTField79.Line[index-1],CRLF)

**IF Substring**(Line,1,2) = “//” THEN

Line = **Substring**(Line,3)

/\* Remove “//” at the beginning of the line \*/

**ENDIF**

**IF** NOT **IsEmpty**(Line) THEN

**IF Length**(PreviousLine)< 50 THEN

/\* Length of PreviousLine must be checked with the “//” including if present \*/

MXString = **Concatenate**(MXString, SPACE, Line)

**ELSE**

MXString = **Concatenate**(MXString, Line)

**ENDIF**

**ELSE**

/\* first line empty, exit the loop. It is not expected to have empty line or line with only “//” \*/

**EXIT Loop**

**ENDIF** /\* END NOT IsEmpty(Line) \*/

**Next** index

/\* Fill in the MX elements \*/

MXCancellationReasonInformation.Reason.Code = “NARR”

MXAdditionalInformation = MXString

**IF length**(MXAdditionalInformation) > 210 THEN

MXAdditionalInformation = **Concatenate**(**Substring**(MXAdditionalInformation, 1,209), “+”)

**ENDIF**

**IF Length**(MXAdditionalInformation) > 105 THEN

MXCancellationReasonInformation.AdditionalInformation[1] = **Substring**(MXAdditionalInformation,1,105)

MXCancellationReasonInformation.AdditionalInformation[2] = **Substring**(MXAdditionalInformation,106)

**ELSEIF Length(**MXAdditionalInformation) > 0 THEN

MXCancellationReasonInformation.AdditionalInformation[1]= MXAdditionalInformation

**ELSE**

/\* Length(MXAdditionalInformation) = 0 after possible clean up (CRLF, “//”) \*/

MXCancellationReasonInformation.AdditionalInformation[1]= “NOTPROVIDED”

/\* With “NARR”, AdditionalInformation is mandatory in MX \*/

**ENDIF**

### 3.3.26 MT\_To\_MXField76

**Name**

MT\_To\_MXField76

**Business description**

The function extracts information from field 76 in MT196/MT296 and translates to camt.029 Cancellation Status and Cancellation Reason Information.

The format of field 76 is expected to be structured as follows

Line 1: /4!c/[Additional information1][[/]AddtionalInformation 2] where 4!c is the cancellation status

Lines 2-NumberOfLines: [//continuation of additional information]

OR :/4c/[Additional information1][[/]AddtionalInformation 2]

On first line /4!c/ must contain one of the following code {CNCL, PDCR, RJCR} which is the status of the cancellation and mandatory in MX. If not present, the translation is stopped in a precondition handled outside of this function.

If a second code /4c/ is present, it will be translated to Additional information (only one reason code is allowed in MX).

AdditionalInformation 1 can contain one of the following MX Reason Code {AC04, AGNT, AM04, ARDT, CUST, INDM, LEGL, NOAS, NOOR, PTNA, RQDA}.

Code “ARPL” has been removed from the CBPR+ list of codes to be aligned with gpi. Should such a code be present in MT, it would be translated to MXAdditionalInformation.

If line 2-NumberOfLines are not structured as described above, they will be treated as pure textual information. It will still be checked if the line starts with “//” in which case “//” will be removed as it would be understood as continuation pattern although the format mixes up structured and unstructured formats.

**Format**

**MT\_To\_MXField76** (MTField76; MXCancellationStatus, MXCancellationStatusReasonInformation)

**Input**

MTField76 : Field76 in MT196/MT296 (6\*35)

**Output**

MXCancellationStatus typed CBPR\_CancellationStatus

MXCancellationStatusReasonInformation typed CancellationStatusReason4

**Preconditions** None.

**Formal description**

/\* Local variables

String76, StatusPattern, AdditionalInfo, TempString,

StructuredPattern,

MXStatus, MXReason, MXReasonList : string

IsStructured : boolean

i: integer \*/

MXReasonList = {AC04, AGNT, AM04, ARDT, CUST, INDM, LEGL, NOAS, NOOR, PTNA, RQDA}

StatusPattern = “/StatusCode/”

where statusCode must be in list{CNCL,PDCR,RJCR}

/\* Extract lines starting with StatusPattern. As per precondition, the Field 76 must start with such a line \*/

String76 = **ExtractLines**(MTField76, StatusPattern,”//”)

**IF Length**(String76)=0 THEN

EXIT function MT\_To\_MXField76

/\* This should not occur due to the Precondition but still coded to have a smooth ending \*/

**ENDIF**

/\* Identify if the field 76 is structured as described in the business description or not \*/

IsStructured = “true”

StructuredPattern = “/4c/”

/\* where 4c is equivalent to XML pattern[0-9A-Z]{1,4} \*/

**For i** = 2 to **NumberOfOccurrences**(MTField76)

/\* check each line in MTField76 starting with Line 2 \*/

**IF Substring**(MTField76.line[i],1,2) = “//” OR

**IF** MTField76.line[i]**starts** with StructuredPattern

/\* check if the line starts with “//” or /4c/ \*/

/\* continue to scan lines \*/

**ELSE**

IsStructured = “false”

Exit loop i

**ENDIF**

**Next i**

**IF** IsStructured

**Call** **SubfunctionStructured76**

/\* Subfunctions described below \*/

**ELSE**

**Call SubfunctionUnStructured76**

**ENDIF**

/\* String76 is the concatenation of the MTField76 lines \*/

/\* Extraction of the Status \*/

MXStatus = **ExtractPattern**(String76, StatusPattern)

/\* Remove “/” \*/

MXStatus = **Substring**(MXStatus,2,4)

/\* Remove the status pattern \*/

String76 = **DeletePattern**(String76, StatusPattern)

**IF WithinList**(**Substring**(String76,1,4),MXReasonList)THEN

MXReason = **Substring**(String76,1,4)

**IF** **Substring**(String76,5,1)= “/” THEN

MXAdditionalInformation = Substring(String76,6)

**ELSE**

MXAdditionalInformation = Substring(String76,5)

**ENDIF**

**ELSE**

MXAdditionalInformation = String76

**IF** MXStatus = “RJCR” THEN

MXReason = “NARR”

**ENDIF**

**ENDIF**

/\* Fill in MX elements \*/

MXCancellationStatus = MXStatus

**IF Length**(MXReason)> 0 THEN

MXCancellationStatusReasonInformation.Reason.Code = MXReason

**ENDIF**

/\* Remove the space added in front of MXAdditionalInformation due to ExtractLines function \*/

MXAdditionalInformation = **TrimLeft**(MXAdditionalInformation,Space)

**IF Length**(MXAdditionalInformation) > 210 THEN

MXAdditionalInformation = **Concatenate**(**Substring**(MXAdditionalInformation,1,209), “+”)

**ENDIF**

**IF Length**(MXAdditionalInformation) > 105 THEN

MXCancellationStatusReasonInformation.AdditionalInformation[1]=

**Substring**(MXAdditionalInformation, 1, 105)

MXCancellationStatusReasonInformation.AdditionalInformation[2]=

**Substring**(MXAdditionalInformation, 106)

**ELSEIF** Length(MXAdditionalInformation) > 0 THEN

MXCancellationStatusReasonInformation.AdditionalInformation[1]= MXAdditionalInformation

**ENDIF**

**SubfunctionStructured76**

/\* At this stage, String76 contains the line(s) with the status code \*/

AdditionalInfo = **DeleteExtractLines**(MTField76, StatusPattern,”//”)

/\* Delete the Status information extracted from String76. Copy remaining string in Additionalinfo \*/

**While Length**(Additionalinfo)> 0 THEN

/\* information is expected to start with /4c/ and might be followed by another /4c/ at next line or “//”. \*/

TempString = **ExtractLines**(AdditionalInfo, /4c/,”//”)

/\* where 4c is equivalent to XML pattern[0-9A-Z]{1,4} \*/

String76 = **Concatenate**(String76, TempString)

/\* if requested , a SPACE could be added in the above concatenation between 2 codes /4c/ and related info. Keeping 2 subfunctions allows such flexibility between structured and unstructured formats\*/

AdditionalInfo = **DeleteExtractLines**(AdditionalInfo,/4c/,”//”)

/\* delete the info extracted to TempString \*/

**End While**

**SubfunctionUnStructured76**

/\* Similar to **SubfunctionNarrative79 \*/**

/\* The translation will just concatenate the lines from Field 76. If the length of a line is less than 35 char, concatenation with the next line will add a space between the 2 lines otherwise the 2 lines are just concatenated without space as it is considered that the next line is the continuation of the previous line. If "//" is used at the beginning of a line (line 2 to 6), it will be deleted as it is interpretated as the continuation of the previous line, although the "//" could be used as normal characters as well (but small risk to be also at the beginning of the line). Note that “//” being used as continuation characters in a free format means that 2 formats are mixed up) \*/

/\* Local variable

String76, PreviousLine, Line : string

index : integer

\*/

/\* Extract MT information. MTField76 is as passed in input parameter at this stage. This extraction is more generic to cater for pure narrative information\*/

String76 = **TrimRight**(MTField76.Line[1], CRLF)

/\* Remove CRLF if present \*/

**For** index = 2 to **NumberOfOccurrences**(MTField76)

/\* For each line in MTField76 repeat \*/

Line = **TrimRight(**MTField76.Line[index],CRLF)

PreviousLine = **TrimRight(**MTField76.Line[index-1],CRLF)

**IF Substring**(Line,1,2) = “//” THEN

Line = **Substring**(Line,3)

/\* Remove “//” at the beginning of the line \*/

**ENDIF**

**IF** NOT **IsEmpty**(Line) THEN

**IF Length**(PreviousLine)< 35 THEN

/\* Length of PreviousLine must be checked with the “//” including if present \*/

String76 = **Concatenate**(String76, SPACE, Line)

**ELSE**

String76 = **Concatenate**(String76, Line)

**ENDIF**

**ELSE**

/\* first line empty, exit the loop. It is not expected to have empty line or line with only “//” \*/

**EXIT Loop**

**ENDIF** /\* END NOT IsEmpty(Line) \*/

**Next** index

### 3.3.27 MT\_To\_MXField79\_2

**Name**

MT\_To\_MXField79\_2

**Business description**

The function extracts information from field 79 in MT292 and translates to camt.058 Cancellation Reason. From a previous translation camt.058 to MT, the code /UETR/ follows by a valid UETR number might be present.

The format of field 79 is either

-narrative 35\*50x

or

-may be structured as follows

Line 1: /4!c/[additional information] where 4!c is the cancellation reason code

Lines 2-35: [//continuation of additional information

[Last Line /UETR/UETRValue]

**In order to reuse as much as possible the translation functions, the same logic will apply to MT\_To\_MXField79 and MT\_To\_MXField79\_2**

An important difference between reason codes in camt.056 and camt.058 : the MT reason codes in MT n92 are all included in camt.056 MX reason codes while the list of MX reason codes in camt.058 is not aligned with the list in MT n92 (\*).

If 4!c is not in the ISO list of reason codes, then Field 79 is translated as such after concatenation of the lines (with the SPACE rule as described in the function MT\_To\_MXField79) to MX AdditionalInformation. In that case Field 79 is treated as a textual field 35\*50 in which “//” starting a line are still deleted. As explained in MT\_To\_MXField79, even in case of Field 79 without any structure, “//” starting a line are removed.

So, /4!c/AdditionalInformation with 4!c not being in the MX Reason Codes is translated as following

MX Reason code = “NARR”

MX AdditionalInformation = “/4!c/AdditionalInformation”

Note that if the continuation sign “//” is not used while /4!c/ is used on the first line (with a valid MX Reason code, restriction due to reusability of the translation function and comment above\*), this will result in a truncation of the next lines as it is not expected in the current translation to have a mix of both formats. If needed and if a CR is requested by customer, this could be improved and aligned with MT\_To\_MXField76.

Additional information can be truncated. This will be indicated with “+” as last character.

**Format**

**MT\_To\_MXField79\_2** (MTField79, MTUETR ; MXOriginalUETR, MXCancellationReason)

**Input**

MTField79 : Field79 in MT292

MTUETR: Block3/EndToEndReference/UniqueEndToEndTransactionReference

**Output**

MXOriginalUETR typed UUIDv4Identifier (located in OriginalItem/UETR)

MXCancellationReason typed NotificationCancellationReason1

**Preconditions** None.

**Formal description**

/\* Local variables

ListOfCode : list of string 4 char

MTCode, MXCode, MXAdditionalInformation : string \*/

/\* MX ISO reason code list in camt.058 \*/

ListOfCode = {DUPL,NOLE,NARR}

Apply the same logic as described in **MT\_To\_MXField79**

# 4 MX to MT Translation Rule Descriptions

This section provides translation rule descriptions for the translation rules for MX to MT Credit Transfer messages. Descriptions are grouped as follows:

* Customer Party Translation Rule Descriptions
* Financial Institution Translation Rule Descriptions
* Other Translation Rule Descriptions

## 4.1 Customer PartyTranslation Rule Description

The translation rule descriptions provided in this section are for translation rules that relate to customer party information.

### 4.1.1 MX\_To\_MTAnyBIC

**Name**

MX\_To\_MTAnyBIC

**Business description**

The function translates an MX AnyBIC party identification to an MT AnyBIC.

**Format**

**MX\_To\_MTAnyBIC** (MXPartyIdentification ; MTAnyBIC)

**Input**

MXPartyIdentification: the entire structure of the MX party identification typed *PartyIdentification135*.

**Output**

MTAnyBIC: AnyBIC identifying the party in the MT format.

**Preconditions**

None.

**Formal description**

**IF**

**IsPresent**(Identification.OrganisationIdentification.AnyBIC)

MTAnyBIC =

Identification.OrganisationIdentification.AnyBIC

**ENDIF**

**Example 1 MX Source:**

An MX Debtor

<Dbtr>

<Nm>Bank ABC</Nm>

<Id>

<OrgId>

<AnyBIC>GKCCBEBB</AnyBIC>

</OrgId>

</Id>

</Dbtr>

**MT Translation:**

:50A:GKCCBEBB

**Example 2 MX Source:**

An MX Creditor

<Cdtr>

<Id>

<OrgId>

< AnyBIC>FOMOUS33</AnyBIC>

</OrgId>

</Id>

</Cdtr>

**MT Translation:**

:59A:FOMOUS33

### 4.1.2 MX\_To\_MTFATFIdentification

**Name**

MX\_To\_MTFATFIdentification

**Business description**

Subfield 1 of an MT target field 50F identifies an ordering customer in line with FATF Special Recommendation VII by providing either an account number or a unique identification of the customer.

The MX party identification typed *PartyIdentification*135 contains an Identification element that offers a choice between an OrganisationIdentification component and a PrivateIdentification component. Organisation identification and private identification are mutually exclusive. There may be multiple occurrences of Other/Identification but only one will be translated to the MT FATF Identification.The other identifications will be ignored.

Depending on the target code, the country or the country with issuer of the identification will be translated from the MX issuer, if available (which will be the case if the original message is an MT message) in order to obtain a correctly formatted FATF compliant identification.

If the Country is not present in the MX Issuer element but is present

-either in the Structured PostalAddress of the Debtor

or in the AddressLine if the MXAddressLine is compliant with the 50F NameAndAddress Structure

or in the CountryOfResidence (only for Private Identification),

the extracted country code will be used to build the FATFidentification. Although the issuer country might not be the same as the Party Postal address country. But a country code is mandatory in FATF identification. IF the country code is not extracted from the MX issuer, a warning will inform the user.

In case it is not possible to create a valid FATFIdentification because the country code is missing, a dummy value “/NOTPROVIDED” is returned in the subfield 1 of field 50F with a warning message.

From the private identification component, the function translates **the first occurring identification** that allows to create a correct FATF ID with an issuer (if requested by the code EMPL, DRLC, CUST). If an issuer is requested and cannot be found (other than the issuer country), a double slash “//” will be introduced between the CountryCode and the Identification

From the Organisation Identification, the following MX codes (schemeName.Code) are first searched “GS1G”, “DUNS” and “TXID”. If none of these codes can be found then the first occurrence of OrganisationID/Other/Identification that allows to create a valid FATF ID will be translated.

IF Organisation ID is present with scheme code “TXID” and “NOTPROVIDED” in Identification, these values are not translated. They are dummy values filled to meet UG rules (ie Account or Other ID must be present) in MT to MX translation.

There are 3 cases to analyse:

Case 1 : The original message is the ISO message and the OrganisationIdentification.Other is present

Case 2 : The original Message is the ISO message and the PrivateIdentification.Other is present

Case 3 : The original message is a MT with Field 50F and PrivateIdentification.Other is present . The MX Issuer element has the following format resulting from a first conversion MT to MX: ISOCountryCode OR ISOCountryCode/Issuer for Code CUST, DRLC, EMPL. Note that the same convention can be used in MX Issuer which is defined as a string.

**In case scenario 1**,

From the Organisation Identification, the following MX codes (SchemeName.Code) are first searched “GS1G”, “DUNS” and “TXID”. The first 2 codes will be translated with “CUST” code and “GS1G and “DUNS” will be part of the MTIssuer. “TXID” will be translated as such to MTFATF identification. If none of these codes can be found then the first occurrence of OrganisationID/Other/Identification that allows to create a valid FATF ID will be translated.

If the MX code is CUST, TXID or EMPL it will be translated to the same MT code, if the MX code is different from CUST, TXIDand EMPL (eg., SRET, BANK), it will be translated by using the MT code “CUST” and the MX code will be used to build the MT issuer identification together with the MX issuer if present. IF the MX code is CUST or EMPL and the MX issuer is absent, then a double slash “//” will be introduced between the CountryCode and the Identification.

The Truncation of the MTIssuer will happen as described in scenario 2 below.

**In case scenario 1, 2**,

For codes EMPL, CUST and DRLC including issuer information the MT Identifier and the line 8 might not be sufficient with 35+33 = 68 char because the MX issuer and MXIdentification could have together up to max 70 char. So in this case the MX issuer will be truncated and the truncation is calculated to keep the full MXIdentification length.

If the MX Issuer in scenario 1, 2 is absent with codes EMPL, CUST or DRLC , a double slash “//” will be introduced between the CountryCode and the Identification.

In scenario 2 where PrivateID is used with SchemeName.Code having no MT equivalent like “TELE”, it will be translated to “CUST” and the MX scheme Name Code is translated to MTIssuer as done for Organisation Scheme Name Codes described above.

**In case scenario 3,**

It is expected to get a valid format from MT meaning the MXIssuer containing the Issuer country code with in addition the issuer identification separated with “/” for the code EMPL, DRLC and CUST.

If the length of the translated outcome goes beyond the MT line format of 35 characters, then the part that needs to be truncated will be stored in a variable to be written under code “8”in Subfield 2 of field 50F (done by another function). Note that there is no guarantee that a line “8/” will be generated as other elements have a higher priority (see MX\_To\_MTFATFNameAndAddress, MX\_To\_MTFATFNameAndAddress 2).

The following identifications are not translated by this function because:

-they are translated by other functions to different field options or subfields:

OrganisationIdentification: AnyBIC, - PrivateIdentification: DateAndPlaceOfBirth

-PrivateIdentification.Other.SchemeName.Proprietary and OrganisationIdentification. Other.SchemeName.Proprietary are removed in CBPR+

**Format**

**MX\_To\_MTFATFIdentification**(MXPartyIdentification ; MTPartyIdentifier, MTCode8) **Input**

MXPartyIdentification: the entire structure of the MX party identification typed *PartyIdentification135*.

**Output**

MTPartyIdentifier: Subfield 1 of field 50F carrying an FATF compliant party identification of the ordering customer in the MT format (4!a//2!a/27x) or a structure like /34x in case a valid FATFIdentification cannot be generated

MTCode8: continuation of an FATF compliant party identification of the ordering customer to be written in the MT message when the MTPartyIdentifier goes beyond 35 characters (including code). MTCode8 might be empty. If present, it contains already the number “8/”.

**Preconditions**

OrganisationIdentification.Other.SchemeName.Code OR PrivateIdentification.Other.SchemeName.Code is present.

**Formal description**

/\* Local variables - string :

MTFATFId

MTCountryCode

MTSchemeCode

MTIssuer

MTIdentifier

MXCode

MXIssuer

SuccessfulFATF, StructuredAddressIndicator : Boolean \*/

SuccessfulFATF = “false”

/\* Search information from OrganisationIdentification scanning each occurrence of OrganisationIdentification/Other and stopping the search when the first MTFATFId can be built from MXCode GS1G, DUNS or TXID with issuer country code. If none of these codes is present then the first occurrence found from OrgID/Other/Identification with an issuer country will be used to build the FATFID. If issuer country cannot be found, then the postal address/Country will be used with the last occurrence of OrgID/Other/Identification \*/

**IF** **IsPresent**(Identification.OrganisationIdentification.Other)THEN

**Case 1**

**/\* search first for Codes GS1G, DUNS or TXID \*/**

**For i= 1, NumberOfOccurrences(OrganisationIdentification.Other)**

MXCode = Identification.OrganisationIdentification.Other[i].SchemeName.Code

MXIssuer = Identification.OrganisationIdentification.Other[i].Issuer

MTIdentifier =

MXPartyIdentification.Identification.OrganisationIdentification.Other[i].Identification

**IF** **Length**(MXIssuer) > 0 AND **IsCountryCode**(Substring(MXIssuer,1,2)AND **Length**(MXIssuer)= 2)THEN

MTCountryCode = MXIssuer

MXIssuer = “”

**ELSEIF** (**IsCountryCode**(Substring(MXIssuer,1,2))AND

**Substring**(MXIssuer,3,1) = “/” AND **Length**(MXIssuer)> 3) THEN

MTCountryCode = **Substring**(MXIssuer,1,2)

MXIssuer = **Substring**(MXIssuer,4)

**ENDIF**

**IF** MXCode **IsInList**{GS1G,DUNS} THEN

/\* Translate it to MTCode “CUST” and MTIssuer being a concatenation of MX SchemeNameCode and MXIssuer. MTIssuer might be truncated later in the function in order to get the full MTIdentification \*/

MTSchemeCode=”CUST”

**IF** **Length**(MXIssuer)> 0 THEN

MTIssuer = **Concatenate**(MXCode,SPACE,MXIssuer)

**ELSE**

MTIssuer= MXCode

**ENDIF**

**IF** **Length**(MTCountryCode)> 0 THEN

SuccessfulFATF = “true”

EXIT FOR loop

**ENDIF**

**ELSEIF** MXCode **IsInList**{TXID}THEN

MTSchemeCode=MXCode

MTIssuer = “”

**IF** LENGTH(MTCountryCode)> 0 AND MTIdentifier NOT equal to “NOTPROVIDED” THEN

/\* Need to remove the dummy value provided in MT to MX translation \*/

SuccessfulFATF = “true”

EXIT FOR loop

**ENDIF**

**ENDIF** /\* ENDIF MXCode \*/

**Next i**

**Case 2**

**/\* Unsuccessful result for GS1G, DUNS or TXID. Take the first occurrence that allows to create a valid FATF ID and translate it. If no issuer country can be found, the latest occurrence of OrganisationID/Other will be used with PostalAddress/Country if present provided the MXCode is in the ISO list \*/**

**IF** SuccessfulFATF = “false” THEN

**For i= 1, NumberOfOccurrences(OrganisationIdentification.Other)**

**{** MXCode = Identification.OrganisationIdentification.Other[i].SchemeName.Code

MXIssuer = Identification.OrganisationIdentification.Other[i].Issuer

MTIdentifier = MXPartyIdentification.Identification.OrganisationIdentification.Other[i].Identification

/\* Search for issuer country code – remove it from MX Issuer \*/

**IF** **Length**(MXIssuer) > 0 AND **IsCountryCode**(**Substring**(MXIssuer,1,2)AND **Length**(MXIssuer)= 2)THEN

MTCountryCode = MXIssuer

MXIssuer = “”

**ELSE IF** (**IsCountryCode**(**Substring**(MXIssuer,1,2))AND

**Substring**(MXIssuer,3,1) = “/” AND **Length**(MXIssuer)> 3) THEN

MTCountryCode = Substring(MXIssuer,1,2)

MXIssuer = Substring(MXIssuer,4)

**ENDIF**

**IF** MXCode **IsInList**(ISO LIST: ExternalOrganisationIdentification1Code) AND MXCode **NOT** **IsInList**{CUST,EMPL,TXID} THEN

/\* Translate it to MTCode “CUST” and MTIssuer being a concatenation of MX SchemeNameCode and MXIssuer. MTIssuer might be truncated later in the function in order to get the full MTIdentification \*/

MTSchemeCode=”CUST”

**IF** **Length**(MXIssuer)> 0 THEN

MTIssuer = Concatenate(MXCode,SPACE,MXIssuer)

**ELSE**

MTIssuer = MXCode

**ENDIF**

**IF** **Length**(MTCountryCode)> 0 THEN

SuccessfulFATF = “true”

EXIT For loop

**ENDIF**

**ELSEIF** MXCode **IsInList**{CUST,EMPL}

/\*MXCode is InList{CUST,EMPL}\*/

/\* Issuer is expected in MT. \*/

MTSchemeCode=MXCode

**IF IsEmpty**(MXIssuer) THEN

MTIssuer = “NOTPROVIDED”

/\* Value “NOTPROVIDED” tracks the cases where an issuer is expected with another value than the country code already removed.

At the end of the function the case will be handled properly to replace the value “NOTPROVIDED” which consumes space by “//”. This allows as well to replace “//” by another value if changes are requested later on. \*/

**ELSE**

MTIssuer = MXIssuer

**ENDIF**

**IF** **Length**(MTCountryCode)> 0 THEN

SuccessfulFATF = “true”

EXIT For loop

**ENDIF**

**ELSEIF** MXCode **IsInList**{TXID} /\* MXCode is TXID \*/

MTSchemeCode=MXCode

MTIssuer = “”

**IF** **Length**(MTCountryCode)> 0 AND MTIdentifier NOT equal to “NOTPROVIDED”

/\* Need to remove the dummy value provided in MT to MX translation \*/

THEN

SuccessfulFATF = “true”

EXIT For loop

**ENDIF**

**ENDIF** /\* MX Code \*/

**}**

**Next i**

/\* if all occurrences have been scanned without exit, the values of the last one is kept. But it must still be checked that the MXCode is in the ISO list \*/

/\* default value for country if not found in the MX Issuer \*/

**IF** SuccessfulFATF = “false” THEN

**IF** MXCode **IsInList**{TXID} AND AND MTIdentifier Equal to “NOTPROVIDED” THEN

/\* No translation \*/

**ELSEIF** MXCode **IsInList**(ISO LIST: ExternalOrganisationIdentification1Code) THEN

**{IF** **IsPresent**(MXPartyIdentification.PostalAddress.Country) THEN

MTCountryCode = MXPartyIdentification.PostalAddress.Country

SuccessfulFATF = “true”

T12001 /\* Error code described in the error code list \*/

**ENDIF** /\* IsPresent Country \*/

/\* Address Line is exclusive with structures postal address \*/

**IF IsPresent(**MXPartyIdentification.PostalAddress.AddressLine)

**call [MX\_To\_MTAddressLineType(**MXPartyIdentification;StructuredAddressIndicator)]

**IF** StructuredAddressIndicator = “true” **THEN**

/\* search for first line starting with “3/” – Country Code is present \*/

**For i = 1 to NumberOfOccurences(**MXPartyIdentification.PostalAddress.AddressLine)

**IF Substring(**MXPartyIdentification.PostalAddress.AddressLine[i],1,2)= “3/” THEN

MTCountryCode= **Substring(**MXPartyIdentification.PostalAddress.AddressLine[i],3,2)=

SuccessfulFATF = “true”

T12001

Exit loop

**ENDIF** /\* start with “3/” \*/

**Next i**

**ENDIF /\*** StructuredAddressIndicator = “true” **\*/**

**ENDIF** /\* ENDIF Present AddressLine \*/

**}**

**ELSE**

T12010 /\* Error code described in the error code list \*/

**ENDIF**

**ENDIF** /\* ENDIF SuccessfulFATF = “false” \*/

**/**\*ENDIF IF Present Organisation/Identification/Other \*/

/\*OrganisationIdentification and PrivateIdentification are mutually exclusive. PrivateIdentification.Other component is repetitive. Below section analyses PrivateIdentification in order to build a valid FATF Identification\*/

**ELSEIF IsPresent**(Identification.PrivateIdentification.Other)THEN

/\* Search information from PrivateIdentification to build a valid MTFATFID. Search for Issuer country code in Issuer \*/

**For i= 1, NumberOfOccurrences(PrivateIdentification.Other)**

MTSchemeCode = Identification.PrivateIdentification.Other[i].SchemeName.Code

MTIssuer = Identification.PrivateIdentification.Other[i].Issuer

MTIdentifier = Identification.PrivateIdentification.Other[i].Identifier

**IF** MTSchemeCode **IsInList**{ ISO20022 ExternalPersonIdentification1Code} THEN

/\* Search for Issuer Country Code \*/

**IF** **Length**(MTIssuer) > 0 AND **IsCountryCode**(Substring(MTIssuer,1,2)) AND **Length**(MTIssuer)= 2)THEN

MTCountryCode = MTIssuer

MTIssuer = “”

SuccessfulFATF = “true”

EXIT FOR loop

**ELSEIF** (**IsCountryCode**(Substring(MTIssuer,1,2))AND

**Substring**(MTIssuer,3,1) = “/” AND **Length**(MTIssuer)> 3) THEN

MTCountryCode = Substring(MTIssuer,1,2)

MTIssuer = Substring(MTIssuer,4)

SuccessfulFATF = “true”

EXIT FOR loop

**ENDIF**

**ENDIF /\*** ENDIF MTSchemeCode IsInList \*/

**Next i**

/\* if all occurrences have been scanned without exit, the values of the last one is kept \*/

/\* default value for country if not found in the MX Issuer \*/

**IF** SuccessfulFATF = “false” AND MTSchemeCode **IsInList**{ ISO20022 ExternalPersonIdentification1Code} THEN

**IF** **IsPresent**(MXPartyIdentification.PostalAddress.Country) THEN

MTCountryCode = MXPartyIdentification.PostalAddress.Country

SuccessfulFATF = “true”

T12001 /\* Error code described in the error code list \*/

**ELSEIF IsPresent(**MXPartyIdentification.PostalAddress.AddressLine)

/\* Address Line is exclusive with structured postal address \*/

**{ call [MX\_To\_MTAddressLineType(**MXPartyIdentification;StructuredAddressIndicator)]

**IF** StructuredAddressIndicator = “true” **THEN**

/\* search for first line starting with “3/” – Country Code is present \*/

**For i = 1 to NumberOfOccurences(**MXPartyIdentification.PostalAddress.AddressLine)

**IF Substring(**MXPartyIdentification.PostalAddress.AddressLine[i],1,2)= “3/” THEN

MTCountryCode= **Substring(**MXPartyIdentification.PostalAddress.AddressLine[i],3,2)=

SuccessfulFATF = “true”

T12001

Exit loop

**ENDIF** /\* start with “3/” \*/

**Next i**

**ENDIF /\*** StructuredAddressIndicator = “true” **\*/**

**ENDIF** /\* ENDIF Present AddressLine \*/

**}**

**ELSEIF** IsPresent(MXPartyIdentification.CountryOfResidence) THEN

MTCountryCode = MXPartyIdentification.CountryOfResidence

SuccessfulFATF = “true”

T12002 /\* Error code described in the error code list \*/

**ENDIF** /\* ENDIF IsPresent Country \*/

**ENDIF /\* ENDIF** SuccessfulFATF = “false” AND MTSchemeCode **IsInList \*/**

**IF** MTSchemeCode **NOT IsInList**{ ISO20022 ExternalPersonIdentification1Code} THEN

SuccessfulFATF = “false”

T12003 /\* Error code described in the error code list \*/

**ENDIF**

**IF** MTSchemeCode IsInList{ARNU,CCPT,NIDN,SOSE,TXID} THEN

MTIssuer = “”

**ENDIF**

/\* For ISO codes with no MT equivalent, MTSchemeCode = “CUST” and the ISO code is translated in MT Issuer as done for codes in Organisation ID without MT equivalent \*/

**IF** MTSchemeCode **IsInList**{ISO20022 ExternalPersonIdentification1Code}AND MTSchemeCode **NOT IsInList**{ARNU, CCPT, CUST, DRLC, EMPL, NIDN, SOSE, TXID} THEN

**IF** **Length**(MTIssuer)> 0 THEN

MTIssuer = **Concatenate**(MTSchemeCode,SPACE,MTIssuer)

**ELSE**

MTIssuer= MTSchemeCode

**ENDIF**

MTSchemeCode = “CUST”

**ENDIF**

**IF** MTSchemeCode IsInList{CUST,DRLC,EMPL} THEN

**IF** **Length**(MTIssuer) = 0 THEN

MTIssuer = “NOTPROVIDED”

**ENDIF**

**ENDIF**

**ENDIF** /\*IsPresent(Identification.Organisation Identification.Other)\*/

**IF** SuccessfulFATF = “false” THEN

{ MTPartyIdentifier = “/NOTPROVIDED”

T12004 /\* Error code described in the error code list \*/

EXIT Function

}

**ELSE**

/\* SuccessfulFATF = “true”. Create the MT FATF Identification from PrivateIdentification or from Organisation Identification\*/

**IF IsEmpty**(MTIssuer) THEN

MTFATFId = Concatenate(MTSchemeCode, “/”,MTCountryCode,”/”,MTIdentifier)

ELSEIF MTIssuer = “NOTPROVIDED” THEN

/\* Replace the value “NOTPROVIDED” by empty in order to generate “//” between country code and identification \*/

MTIssuer = “”

MTFATFId = **Concatenate**(MTSchemeCode, “/”,MTCountryCode,”/”,MTIssuer, “/”, MTIdentifier)

**ELSE**

MTFATFId = **Concatenate**(MTSchemeCode, “/”,MTCountryCode,”/”,MTIssuer, “/”, MTIdentifier)

**ENDIF** /\* EndIF IsEmpty \*/

/\*Building MTIdentifier (subfield 1)and MTCode8 when needed, if SuccessfulFATFID = “True”.

Length restriction MTFATFId to 35x. If longer, the truncated part is written in MTCode8 (used for translation to field 50F - Subfield 2, called in another function described elsewhere). In the latter case the truncated part of the MTFATFId itself will be restricted to 33 characters \*/

/\* For codes EMPL, CUST and DRLC including issuer information the MT Identifier and the line 8 might not be sufficient with 35+33 = 68 char while MX issuer and MXIdentification could have max 70 char. So in this case the MX issuer will be truncated after 24 char and sign “+” added to guarantee no truncation of the MXIdentification \*/

**IF MTSchemeCode IsInList{EMPL,CUST,DRLC}** THEN

**IF** **Length**(MTFATFId) > 68 THEN

/\* This is only possible if MTIssuer is not empty \*/

/\*Truncate MTIssuer char and add truncate sign “+”. Rebuild MTFATFId\*/

/\*Calculate the length of MT issuer in order to keep the full MTIdentifier \*/

/\* Structure is Code/CountryCode/MTIssuer/Id with length of 35 +33 = 68 char out of which the fixed length with 3 slashes is 9 char => left room is 68-9 = 59 char\*/

IssuerLength = 59-**Length**(MTIdentifier)

MTIssuer = **Concatenate**(**Substring**(MTIssuer, 1, IssuerLength-1), “+”)

MTFATFId = **Concatenate**(MTSchemeCode, “/”,MTCountryCode,”/”,MTIssuer, “/”, MTIdentifier)

**ENDIF** /\* ENDIF LENGTH(MTFATFId) > 68 \*/

**ENDIF**

/\* Copy MTFATFId to MTIdentifier and build MTCode8 if needed \*/

/\* MTFATId length is less or equal to MT Identifier Length + MTCode8 Length which allows up to 68 char \*/

**IF** **Length**(MTFATFId) > 35 THEN

MTPartyIdentifier = **Substring**(MTFATFId, 1, 35)

MTCode8 = **Concatenate**(“8/”, **Substring**(MTFATFId, 36))

**ELSE**

MTPartyIdentifier = MTFATFId

**ENDIF**

**ENDIF** /\* SuccessfulFATFID \*/

### 4.1.3 MX\_To\_MTFATFNameAndAddress

**Name**

MX\_To\_MTFATFNameAndAddress

**Business description**

Subfield 2 of an MT target field 50F identifies an ordering customer in line with FATF Special Recommendation VII with one of the following combinations:

* name and address
* name and date and place of birth
* name and national identity number
* name and LEI
* name and customer number

This function will be used when BIC of the Debtor is absent and a structured address is present, at minimum the country (in camt messages, Name is not mandatory)

In absence of an account information, one occurrence of OrganisationId or Private Id (which are exclusive) is already used to build the FATF Identification.

Subfield 2 of Field 50F will be filled in with the following priority depending on the room left:

1/Name (max 2 occurrences)

2/ StreetName line (as per MX\_To\_MTPartyNameAndStructuredAddress)

3/CountryCode line (as per MX\_To\_MTPartyNameAndStructuredAddress)

4/ BirthDate (Private identification)

5/ Birth Country and Place of Birth (City) (Private identification)

6/ CUST information (PrivateIdentification) OR 6/LEI (OrganisationIdentification)

7/NIDN information (Private identification)

8/ Continuation of PartyIdentifier (if needed and room left).

Number 1/ and 3/ are mandatory and Number 4/ and 5/ if present, must be present together or none of them.

Number 6/CUST and 7/NIDN will not be translated if already used to build the FATF Identification.

If Account is absent and if the PartyIdentifier must be completed, number “8/” will be used if room left.

LEI priority. The function MX\_To\_MTPartyNameAndStructuredAddress reserves one line for LEI if present. The presence of the LEI is taken into account to calculate the room left for the translation of the structured postal address (2/StreetLine and 3/CountryLine). In other words, LEI, if present must be translated due to the reserved line.

As LEI can only be present in OrganisationID, it cannot conflict with the other numbers (4/, 5/, 6/ and 7/) extracted from Private ID per construction. LEI has priority on “8/” in case it is present.

.

**Format**

**MX\_To\_MTFATFNameAndAddress**(MXPartyIdentification, MXPartyAccount ;

MTFATFNameAndAddress)

**Input**

MXPartyIdentification: the entire structure of the MX party identification typed *PartyIdentification135*.

MXPartyAccount : account related to the party for which the identification is passed in MXPartyIdentification and typed *CashAccount38*.

**Output**

MTNameAndAddress: party identification structured as per field 50F - Subfield 2 format (up to 4 lines of 1!n/33x where n is a number between 1 and 8).

**Preconditions**

Name is present

**Formal description**

/\*MTNameAndAddress is defined by 4 lines of 35 characters.

MTNameAndAddress[1] indicates the first line\*/

/\*Throughout the function, if translation of the source

component is spread over more than one line of the 4\*35x format, a Carriage Return Line Feed (*CRLF*) will be added

between consecutive lines to comply with the format of an MT field with multiple lines\*/

/\* The function MX\_To\_MTFATFNameAndAddress is called when BIC of the party and AddressLine are absent. In CBPR+, if the party BIC is absent then the Name is mandatory in pacs (but not in camt.054) \*/

/\* The function MX\_To\_MTFATFIdentification below is called because the output string MTCode8 is needed as input string for this function (the function MX\_To\_MTFATFIdentification is described elsewhere and is only called in absence of an account number \*/

/\*Local variables

MTCode8: string

NumberOfAvailableLines : integer

MTTable[]: table of strings

MTCode5, MTCode6, MTCode7, MTCode8, MTPartyIdentifier, MXSchemeCode, AdditionalInformation, IssuerCountryCode : string

MTCode6Indicator, MTCode7Indicator : boolean \*/

/\*Fill in lines with Number 1/, 2/ and 3/. At a minimum Name must be present\*/

**MX\_To\_MTPartyNameAndStructuredAddress**(MXPartyIdentification; MTFATFNameAndAddress)

/\* Information will be extracted to build line with Number 4/, 5/, 6/, 7/ and 8/ \*/

/\* MTTable[1] => 8/MTCode8;

MTTable[2]=> 4/DateOfBirth;

MTTable[3]=> 5/Country and Place of birth;

MTTable[4] => 6/CUST or 6/LEI;

MTTable[5] => 7/NIDN; \*/

/\* Extract information for Number 8/ - Stored in MTTable [1]\*/

**IF** MXPartyAccount **IsAbsent** THEN

**MX\_To\_MTFATFIdentification**(MXPartyIdentification ; MTPartyIdentifier, MTCode8)

/\* as built, MTCode8 contains already 8/ if not emppty \*/

**IF Length**(MTCode8) > 35 THEN

MTCode8 = **Concatenate**(**Substring**(MTCode8, 1, 34), “+”))

**ENDIF**

MTTable[1] = MTCode8

**ENDIF** /\* MXPartyAccount \*/

/\* Extract 4/DateOfbirth; 5/Country and Place of birth\*/

**IF** MXPartyIdentification.Identification.PrivateIdentification.DateAndPlaceOfBirth **IsPresent** THEN

**MX\_To\_MTBirthInformation**(MXPartyIdentification; MTCode4, MTCode5)

MTTable[2]= MTCode4

MTTable[3]= MTCode5

**ENDIF**

/\* Extract information for 6/CUST or 7/NIDN if not present in MTPartyIdentifier \*/

**IF**  MXPartyIdentification.Identification.PrivateIdentification.Other IsPresent THEN

**MX\_To\_MTFATFID\_CUST\_NIDN**(MXPartyIdentification,MXPartyAccount, MTPartyIdentifier,; MTCode6, MTCode7)

MTTable[4]= MTCode6

MTTable[5]= MTCode7

**ENDIF**

/\* PrivateIdentification and OrganisationIdentification are exclusive \*/

**IF**  MXPartyIdentification.Identification.OrganisationIdentification.LEI IsPresent THEN

MTCode6 = MXPartyIdentification.Identification.OrganisationIdentification.LEI

/\* If available. If not, use default = GLEIF headquarters in Basel, i.e.CH. LEIC is always fixed. \*/

**IF** MXPartyIdentification.PostalAddress.Country IsPresent THEN

IssuerCountryCode = MXPartyIdentification.PostalAddress.Country

**ELSE**

IssuerCountryCode = “CH”

**ENDIF**

MTCode6 = Concatenate(“6/”,IssuerCountryCode,”/LEIC/”, MTCode6)

MTTable[4]= MTCode6

**ENDIF** /\* Organisation LEI present \*/

/\* Check if information is present to be copied to 50F subfield 2 \*/

**IF** (**Length**(MTTable[1])=0 AND **Length**(MTTable[2])=0 AND **Length**(MTTable[3])=0 AND **Length**(MTTable[4])=0 AND **Length**(MTTable[5]))=0 THEN

/\*No additional information to copy \*/

EXIT function

**ENDIF**

/\* fill in Structured address with 4/, 5/ 6/ 7/ and 8/ if information is present and if room left. After the structured Name and Address, Birth information is filled if room left, 2 lines are needed. Then Number 6/ and/or 7/ and Number 8/.

All information has been truncated if needed in the different functions above. MTTable[] can be empty. So it is needed to check if information is present\*/

**IF** **ReturnFirstLineEmpty**(MTFATFNameAndAddress,4)= 0 THEN

/\* there is information present but no room left \*/

MissingInformation\_Flag = “True”

Exit function

**ELSE**

/\*For developers only. Implementation below may be different but should reflect the same logic. The function should be easily maintained in case there is a CR in the priority of the element to be used to fill Field 50F subfiled 2 from MTTable[1], MTTable[4]and MTTable[5] \*/

NumberOfAvailableLines = 4 - **ReturnFirstLineEmpty**(MTFATFNameAndAddress,4)+1

**IF** NumberOfAvailableLines = 1 THEN

/\* Candidates are 6/CUST or 6/LEI => MTTable[4] or 7/NIDN => MTTable[5] or MTCode8 (party identifier remaining part) => MTTable[1] \*/

**IF** **Length**(MTTable[4])> 0 THEN

AdditionalInformation = MTTable[4]

**ELSEIF** **Length**(MTTable[5])> 0 THEN

AdditionalInformation = MTTable[5]

**ELSEIF** **Length**(MTTable[1])> 0 THEN

AdditionalInformation = MTTable[1]

**ENDIF** /\* IF LENGTH(MTTable[4])> 0 \*/

**IF** **Length** (AdditionalInformation) > 0 THEN

**AppendToNextLine**(AdditionalInformation, MTFATFNameAndAddress))

**ENDIF**

**IF** at least 2 elements amongst {MTTable[4], MTTable[5], MTTable[1]}are present OR IF (MTTable[4] AND MTTAble[5]) are present THEN

/\* only one has been copied \*/

Flag\_MissingInformation = True

**ENDIF**

**ENDIF** /\* NumberOfAvailableLines = 1 \*/

**IF** NumberOfAvailableLines = 2 THEN

/\* Apply the same logic including the Flag\_Missing information if needed. Candidates are by priority and if the elements are not empty

**First** MTTable[2] AND MTTable[3] must be present together or none of them

Then 6/CUST or 6/LEI => MTTable[4], if present; 7/NIDN => MTTable[5]if present or MTCode8 (party identifier remaining part) => MTTable[1]if present \*/

/\*For each filled line, max 2 \*/

**AppendToNextLine…**

**ENDIF** /\* NumberOfAvailableLines = 2 \*/

**IF** NumberOfAvailableLines = 3 THEN

/\* Apply the same logic including the Flag\_Missing information if needed. Candidates are by priority and if the elements are not empty

**First** MTTable[2] AND MTTable[3] must be present together or none of them

Then 6/CUST or 6/LEI => MTTable[4], if present; 7/NIDN => MTTable[5]if present or MTCode8 (party identifier remaining part) => MTTable[1]if present \*/

/\*For each filled line, max 3 \*/

**AppendToNextLine…**

**ENDIF** /\* NumberOfAvailableLines = 3 \*/

/\*NumberOfAvailableLines = 4 is not possible as at least 1 line is used for the Name which is mandatory \*/

**ENDIF** /\* ReturnFirstLineEmpty = 0 \*/

### 4.1.4 MX\_To\_MTFATFNameAndAddress2

**Name**

MX\_To\_MTFATFNameAndAddress2

**Business description**

The function is used to create subfield2 in 50F when MXAddressLine is structured and contains lines starting with “2/” or “3/” coming from previous MT 50F or original MX with structured postal address translated to MT. Depending on the room left, if LEI is present then the Name length is limited to 33 char else Name length is limited to 66 char. Then if there are still 2 lines left, it is checked if birth date and birth place information can be translated. As PrivateIdentification and OrganisationId are exclusive, LEI and PrivateId are exclusive.

Then If PrivateIdentification/Other is present, it is checked if line “6/” can be built, then if line “7/” can be built

If there is 1 line left and no account present, it is checked whether FATF Identification needs to be continued with number “8/”.

OrganistionIdentification/Other will be ignored in subfield 2.

**Format**

**MX\_To\_MTFATFNameAndAddress2**(MXParty,MXPartyAccount; MTNameAndAddress)

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

MXPartyAccount : the entire structure of the MXPartyAccount typed *cashAccount38*

**Output**

MTNameAndAddress with the format defined in 50F subfield 2

**Preconditions**

MX Address Line is present and structured with Number “2/” and “3/”

**Formal description**

/\* Local variables

MTNameAndAddressTable [ ] : table of string

AddressLineNumberOfOccurrences, Index, i : integer

MXName : string

MXLEI, MXCountryCode : string

MTBirhthDate, MTBirthPlace : string

MTPartyIdentifier, MTCode8, MTCode6, MTCode7 : string \*/

**IF** **Length**(MXParty.Name) = 0 THEN

MXName = “NOTPROVIDED”

T12009

**ELSE**

MXName = MXParty.Name

**ENDIF**

AddressLineNumberOfOccurrences = **NumberOfOccurrences**(MXParty.PostalAddress.AddressLine)

**IF Length**(MXParty.Identification.OrganisationIdentification.LEI) > 0 THEN

MXLEI = MXParty.Identification.OrganisationIdentification.LEI

/\* Search for the Country code \*/

**For** index = 1 to AddressLineNumberOfOccurrences

**IF** **IsPresentPattern**(**Substring**(MXParty.PostalAddress.AddressLine[index],1,2), “3/”)

THEN

/\* country code is expected in the first “3/” occurrence \*/

**IF IsCountry**(**Substring**(MXParty.PostalAddress.AddressLine[index], 3,2) THEN

MXCountryCode = **Substring**(MXParty.PostalAddress.AddressLine[index], 3,2)

Exit loop

**ENDIF**

**ENDIF**

**Next** index

**END** loop

/\* Fill in with 6/CountryCode/LEIC/LEIIdentifier \*/

**IF** **Length**(MXCountryCode) = 0 THEN

MXCountryCode = “CH”

**ENDIF**

MTLEI = **Concatenate**(“6/”,MXCountryCode,”/LEIC/”, MXLEI)

**ENDIF** /\* Length(LEI) \*/

**IF** AddressLineNumberOfOccurrences = 3 THEN

/\* Name length is limited to 1 line \*/

**IF** **Length(**MXName) > 33 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,32),”+”)

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

**ENDIF**

index = 1

**For i** = 1 to AddressLineNumberofOccurrences

index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

**IF Length**(MXLEI) > 0 THEN

Flag\_MissingInformation = “True”

**ENDIF**

**ELSEIF** AddressLineNumberOfOccurrences = 2 THEN

/\* IF LEI is present, then Name is limited to 1 line else Name can use 2 lines \*/

**IF** **Length**(MXLEI) > 0THEN

**IF** **Length**(MXName) > 33 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,32),”+”)

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

**ENDIF**

index = 1

**For i** = 1 to AddressLineNumberofOccurrences

index = index + 1

MTNameAndAddressTable[index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

index = index + 1

MTNameAndAddressTable[index] = MTLEI

**ELSE** /\* AddressLineNumberOfOccurrences = 2, no LEI, Name can use up to 2 lines \*/

**IF** **Length**(MXName) > 66 Then

MTNameAndAddressTable [1] = **Concatenate**(“1/”, Substring(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, Substring(MXName, 34,32), “+”)

Index = 2

**ELSEIF** **Length**(MXName) > 33

MTNameAndAddressTable [1] = **Concatenate**(“1/”, Substring(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, Substring(MXName,34))

Index = 2

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

Index = 1

**ENDIF**

**For i** = 1 to AddressLineNumberOfOccurrences

index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

/\* ENDIFAddressLineNumberOfOccurrences = 2 \*/

**ELSEIF** AddressLineNumberOfOccurrences = 1

/\* meaning no line 2/ only 3/ in AddressLine\*/

/\* Name can use up to 2 lines \*/

/\* 1 line can be used by LEI if present \*/

**IF** **Length**(MXName) > 66 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, Substring(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, Substring(MXName, 34,32), “+”)

Index = 2

**ELSEIF Length**(MXName) > 33 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, Substring(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, Substring(MXName,34)

Index = 2

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

Index = 1

**ENDIF**

**For i** = 1 to AddressLineNumberOfOccurrences

Index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

**IF Length**(MXLEI) > 0 THEN

index = index + 1

MTNameAndAddressTable [index] = MTLEI

**ENDIF**

**ENDIF**  /\* IF AddressLineNumberOfOccurrences = 3 \*/ /\* If there are still 2 lines left, birthday date and place could be used to fill MTNameAndAddress. OrganisationId and PrivateId are exclusive. So no conflict in priority between LEI priority and elements from PrivateID \*/

**IF** MXParty.Identification.PrivateIdentification.DateAndPlaceOfBirth **IsPResent** THEN

**MX\_To\_MTBirthInformation**(MXParty; MTBirhthDate, MTBirthPlace)

/\* MTBirthDate and MTBirthPlace are already formatted with the number \*/

/\* birth information can be added to MT Structure \*/

index = index + 1

MTNameAndAddressTable [index] = MTBirthDate

index = index + 1

MTNameAndAddressTable [index] = MTBirthPlace

**ELSE**

ReportMissingInformation **ENDIF** /\* index \*/

**ENDIF /\*** DateAndPlaceOfBirth information is present \*/

/\* Search if there is MTCode8 to continue FATF Identification. Last element to be added if space left. But need to extract MTPartyIdentifier at this point to check presence of CUST and NIDN, in which case no translation to subfield2 if present in PartyIdentifier \*/

**IF** MXPartyAccount IsAbsent THEN

**MX\_To\_MTFATFIdentification**(MXParty ; MTPartyIdentifier, MTCode8)

/\* as built, MTCode8 contains already 8/ if not emppty \*/

**IF Length**(MTCode8) > 0 THEN

**IF** **Length**(MTCode8) > 35 THEN

MTCode8 = **Concatenate**(**Substring**(MTCode8, 1, 34), “+”))

**ENDIF**

**ENDIF** /\* Length(MTCode8) \*/

**ENDIF** /\* MXPartyAccount Present \*/

/\* Extract information for 6/CUST or 7/NIDN if not present in MTPartyIdentifier \*/

**IF** MXParty.Identification.PrivateIdentification.Other IsPresent THEN

**MX\_To\_MTFATFID\_CUST\_NIDN**(MXParty,MXPartyAccount, MTPartyIdentifier; MTCode6, MTCode7)

**IF** Length(MTCode6) > 0 THEN

**IF** index < 4 THEN

index = index + 1

MTNameAndAddressTable [index] = MTCode6

**ELSE**

ReportMissingInformation

**ENDIF**

**ENDIF** /\* Length(MTCode6) \*/

**IF** Length(MTCode7) > 0 THEN

**IF** index < 4 THEN

index = index + 1

MTNameAndAddressTable [index] = MTCode7

**ELSE**

ReportMissingInformation

**ENDIF**

**ENDIF** /\* Length(MTCode7) \*/

**ENDIF /\* PrivateIdentification.Other is present \*/**

/\* Copy MTCode8 if present \*/

**IF** Length(MTCode8) > 0 THEN

**IF** index < 4 THEN

index = index + 1

MTNameAndAddressTable [index] = MTCOde8

**ELSE**

ReportMissingInformation

**ENDIF**

**ENDIF** /\* Length(MTCode8) \*/

/\* Copy information to MT structure \*/

**For i= 1** to index

**AppendToNextLine**(MTNameAndAddressTable [i],MTNameAndAddress)

**Next i**

### 4.1.5 MX\_To\_MTFATFID\_CUST\_NIDN

**Name**

MX\_To\_MTFATFID\_CUST\_NIDN

**Business description**

The function searches information with scheme code CUST and NIDN from PrivateIdentification.Identification.OtherIdentification, then extracts the information and builds a valid structure for Number 6/ and Number 7/ in 50F subfield 2 provided the information is not already present in the FATF identification (subfield 1 field 50F). This number can also be present if subfield 1 is an account. If the MX issuer is absent, a double slash “//” is used between the country code and the identification with the code CUST.

The Country Code is searched from MXIssuer (if payment was originated in MT with FATFIdentication from Field 50F) or from PostalAddress.CountryCode or from Structured AddressLine or from CountryOfResidence. If no country code can be found, the line 6/ and 7/ are not created.

**Format**

**MX\_To\_MTFATFID\_CUST\_NIDN**(MXParty,MXAccount, MTPartyIdentifer; MTCUST, MTNIDN)

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

MXAccount : entire Party account structure typed CashAccount38

MTPartyIdentifier : string

**Output**

Structured information 6/CUST structure ; 7/ NIDN structured as defined for subfield 2 in 50F

**Preconditions**

MXParty.PrivateIdentification.Other is present

**Formal description**

/\* Local variables

i, Index6, Index7 : integer

MXSchemeCode, MXIssuer, MXIdentifier :string

MTCode6Indicator, MTCode7Indicator, SuccessfulFATF : boolean

MTCountryCode, MTIssuer : string \*/

MTCode6Indicator = false

MTCode7Indicator = false

SuccessfulFATF = false

/\* search for CUST code \*/

**For** i=1 to **NumberOfOccurrences**(MXPartyIdentification.Identification.PrivateIdentification.Other)

MXSchemeCode = MXPartyIdentification.Identification.PrivateIdentification.Other[i].SchemeName.Code

**IF** MXSchemeCode = “CUST” THEN

MTCode6Indicator = True

Index6 = i

Exit loop

**ENDIF**

**END** **LOOP**

/\* search for NIDN code \*/

**For** i=1 to **NumberOfOccurrences**(MXPartyIdentification.Identification.PrivateIdentification.Other)

MXSchemeCode = MXPartyIdentification.Identification.PrivateIdentification.Other[i].SchemeName.Code

**IF** MXSchemeCode = “NIDN” THEN

MTCode7Indicator = True

Index7 = i

Exit loop

**ENDIF**

**END LOOP**

**/\* build Line 6/CUST line \*/**

**IF** MTCode6Indicator is **True** AND **(**MXAccount **IsPresent** OR (MXAccount **IsAbsent** AND MTPartyIdentifier **NOT Equal** to “/NOTPROVIDED” AND **Substring**(MTPartyIdentifier, 1, 4)Is **Not Equal** “CUST”)**)** THEN

/\* Presence of MTPartyIdentifier (generated only if MXAccount is absent)and MX account are exclusive. IF CUST is present in MTPartyIdentifier the 6/CUSt line is not built \*/

/\* 2 cases are possible as described in function

MX\_To\_MTFATFNameAndAddress. Either the original payment is originated in MX (Case 1) or MX is the result of a previous translation and the original payment was originated in MT (Case 2). Both scenarios must be analysed\*/

MXIssuer = MXPartyIdentification.Identification.PrivateIdentification.Other[Index6].Issuer

MXIdentifier = MXPartyIdentification.Identification.PrivateIdentification.Other[Index6].Identification

/\* If Private.OtherId is present then SchemeName code and Identification are mandatory in CBPR+. Issuer is optional. If Issuer is not present, dummy value NOTPROVIDED is used \*/

/\* scenario case 2 \*/

**IF** **IsCountryCode**(**Substring**(MXIssuer,1,2))AND

**IF** **Substring**(MXIssuer,3,1) = “/” AND **Length**(MXIssuer)> 3 THEN

MTCountryCode = **Substring**(MXIssuer,1,2)

MTIssuer = **Substring**(MXIssuer,4)

SuccessfulFATF = “true”

/\* scenario case 1 – Handles as well the case where MXIssuer is filled in with only the country code \*/

**ELSEIF**

**Length**(MXIssuer) > 0 AND **IsCountryCode**(**Substring**(MXIssuer,1,2)) AND **Length**(MXIssuer)= 2)THEN

MTCountryCode = MXIssuer

MTIssuer = “”

SuccessfulFATF = “true”

**ELSE**

**{**

**IF** **Length**(MXIssuer)> 0 THEN

MTIssuer = MXIssuer

**ENDIF**

**IF IsPresent**(MXParty.PostalAddress.CountryCode)THEN

MTCountryCode= MXParty.PostalAddress.CountryCode

SuccessfulFATF = “true”

**ElSEIF IsPresent**(MXParty.PostalAddress.AddressLine) THEN

**SubfunctionExtractCountryCodeAddressLine**(MXPartyIdentification,SuccessfulFATF;MTCountryCode)

**/\* subfunction defined below \*/**

**ENDIF**

**IF** IsPresent(MXParty.CountryOfResidence)AND SuccessfulFATF = “false”

MTCountryCode = MXParty.CountryOfResidence

SuccessfulFATF = “true”

**ENDIF**

**}**

**ENDIF**

/\* Default value for MTIssuer \*/

**IF** **Length**(MTIssuer) = 0 THEN

MTIssuer = ””

/\* Empty string to generate “//” between Country code and Identification \*/

**ENDIF**

**IF** SuccessfulFATF = “True” THEN

MTCUST = **Concatenate**(“6/”, MTCountryCode, ”/”, MTIssuer, ”/”, MXIdentifier)

**IF** **Length**(MTCUST)> 35 THEN

MTCUST = **Concatenate**(**Substring**(MTCUST,1,34),”+”))

**ENDIF**

**ELSE**

/\* no country code to create valid Number 6/ \*/

Flag\_MissingInformation= True

T12005 /\* Error code described in the error code list \*/

**ENDIF**

**ENDIF /\*build line 6 \*/**

**/\* Build Line 7/ \*/**

SuccessfulFATF = “false”

**IF** MTCode7Indicator is **True** AND **(**MXAccount IsPresent OR (MXAccount **IsAbsent** AND MTPartyIdentifier **NOT Equal** to “/NOTPROVIDED” AND **Substring**(MTPartyIdentifier, 1, 4)**Is Not Equal** “NIDN”)**)** THEN

/\* Presence of MTPartyIdentifier (generated only if MXAccount is absent) and MX account are exclusive. IF NIDN is present in MTPartyIdentifier the 7/NIDN line is not built \*/

/\* 2 cases are possible as described in function

MX\_To\_MTFATFNameAndAddress. Either the original payment is originated in MX (Case 1) or MX is the result of a previous translation and the original payment was originated in MT (Case 2). Both scenarios must be analysed\*/

MXIdentifier = MXPartyIdentification.Identification.PrivateIdentification.Other[Index7].Identification

/\* If Private.OtherId is present then SchemeName code and Identification are mandatory in CBPR+. Issuer is optional. If Issuer is not present, dummy value NOTPROVIDED is used \*/

/\* scenario case 2 \*/

**IF** **IsCountryCode**(Substring(MXIssuer,1,2))THEN

MTCountryCode = **Substring**(MXIssuer,1,2)

SuccessfulFATF = “true”

/\* scenario case 1 \*/

**ELSE**

**{**

**IF** **IsPresent**(MXParty.PostalAddress.CountryCode)THEN

MTCountryCode= MXParty.PostalAddress.CountryCode

SuccessfulFATF = “true”

**ElSEIF IsPresent**(MXParty.PostalAddress.AddressLine) THEN

**SubfunctionExtractCountryCodeAddressLine**(MXPartyIdentification,SuccessfulFATF;MTCountryCode)

/\* subfunction defined below \*/

**ENDIF**

**IF IsPresent**(MXParty.CountryOfResidence) AND SuccessfulFATF = “false”

MTCountryCode = MXParty.CountryOfResidence

SuccessfulFATF = “true”

**ENDIF**

**}**

**ENDIF**

**IF** SuccessfulFATF = “True” THEN

MTNIDN = **Concatenate**(“7/”,MTCountryCode,”/”, MXIdentifier)

**IF** **Length**(MTNIDN)> 35 THEN

MTNIDN = **Concatenate**(**Substring**(MTNIDN,1,34),”+”))

**ENDIF**

**ELSE**

/\* no country code to create valid Number 7/ \*/

Flag\_MissingInformation= True

T12005 /\* Error code described in the error code list \*/

**ENDIF**

**ENDIF / build line 7/**

**SubfunctionExtractCountryCodeAddressLine**(MXPartyIdentification, SuccessfulFATF,CountryCode)

StructuredAddressIndicator: boolean

**MX\_To\_MTAddressLineType(**MXPartyIdentification;StructuredAddressIndicator)]

**IF** StructuredAddressIndicator = “true” **THEN**

/\* search for first line starting with “3/” – Country Code is present \*/

**For i = 1 to NumberOfOccurences(**MXPartyIdentification.PostalAddress.AddressLine)

**IF Substring(**MXPartyIdentification.PostalAddress.AddressLine[i],1,2)= “3/” THEN

CountryCode= **Substring(**MXPartyIdentification.PostalAddress.AddressLine[i],3,2)=

SuccessfulFATF = “true”

T12001

Exit loop

**ENDIF** /\* start with “3/” \*/

**Next i**

**ENDIF /\*** StructuredAddressIndicator = “true” **\*/**

### 4.1.6 MX\_To\_MTUltimateParty

**Name**

MX\_To\_MTUltimateParty

**Business description**

The function extracts the UltimateParty information in order to return a string following the below pattern and priority, dependent on the presence of elements.

If BIC is present, it will be translated and other elements are ignored.

Then Name/Country[/TownName] OR Name/OtherId OR Name OR OtherId

Note that only the first occurrence of OtherId is translated.

**Format**

MX\_To\_MTUltimateParty(MXUltimateParty; MTUltimateParty)

**Input**

MXUltimateParty typed PartyIdentification135

**Output**

MTUltimateParty : string

**Preconditions**

None

**Formal description**

/\* Local variables:

MXName, MXCountry, MXTownName, MXIdentification, MXBIC : string \*/

MXBIC = MXUltimateParty.Identification.OrganisationIdentification.AnyBIC

MXName = MXUltimateParty.Name

MXCountry = MXUltimateParty.PostalAddress.Country

MXTowName = MXUltimateParty.PostalAddress.TownName

/\* Translate BIC if present and ignore the other elements \*/

**IF Length**(MXBIC)> 0 THEN

MTUltimateParty = MXBIC

EXIT function

**ENDIF**

/\* IF BIC is absent, search for the other elements \*/

**IF** MXUltimateParty.Identification.OrganisationIdentification.Other[1].Identification **IsPresent** THEN

MXIdentification = MXUltimateParty.Identification.OrganisationIdentification.Other.Identification

**ELSEIF** MXUltimateParty.Identification.PrivateIdentification.Other.Identification **IsPresent** Then

MXIdentification = MXUltimateParty.Identification.PrivateIdentification.Other[1].Identification

**ENDIF**

**Case 1** **Length**(MXName) > 0

**IF Length**(MXCountry) > 0 THEN

**IF** **Length**(MXTownName) > 0 THEN

MTUltimateParty =

**Concatenate**(MXName, “/”, MXCountry, “/”, MXTownname)

**ELSE**

MTUltimateParty =

**Concatenate**(MXName, “/”, MXCountry)

**ENDIF**

**ELSE** /\* no Country but Name is present \*/

**IF Length**(MXIdentification) > 0 THEN

MTUltimateParty = **Concatenate**(MXName, “/”, MXIdentification)

**ELSE**

MTUltimateParty = MXName

**ENDIF**

**ENDIF** /\* IF Length(MXCountry) > 0 \*/

**Case 2** **Length**(MXName) = 0

**IF** **Length(**MXIdentification) > 0 THEN

MTUltimateParty = MXIdentification

**ELSE** MTUltimateParty = “”

**ENDIF**

### 4.1.7 MX\_To\_MTBirthInformation

**Name**

MX\_To\_MTBirthInformation

**Business description**

The function extracts the birth information from MX Party and format it with the MT structured format described in Field 50F. If the length of Place of birth is longer than 30 characters, it will be truncated and sign “+ “ is added to indicate truncation.

**Format**

**MX\_To\_MTBirthInformation**(MXParty ; MTBirthDate, MTCountryandPlaceOfBirth)

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

**Output**

Structured information 4/MTBirthDate; 5/ MTCountryandPlaceOfBirth

**Preconditions**

MXParty.PrivateIdentification.DateAndPlaceOfBirth is not empty

**Formal description**

/\* Local variable

MTBirthDate, MXBirthDate, MXBirthPlace, MXBirthCountry : string \*/

MXBirthDate = MXPartyIdentification.Identification.PrivateIdentification.DateAndPlaceOfBirth.BirthDate

MXBirthPlace = MXPartyIdentification.Identification.PrivateIdentification.DateAndPlaceOfBirth.CityOfBirth

MXBirthCountry = MXPartyIdentification.Identification.PrivateIdentification.DateAndPlaceOfBirth.CountryOfBirth

**MX\_To\_MTDate**(MXBirthDate, MTBirthDate)

MTBirthDate = **Concatenate**(“4/”, MTBirthDate)

**IF Length**(MXBirthPlace)> 30 THEN

MTBirthPlace = Concatenate(Substring(MXBirthPlace,1,29), “+”)

**ELSE**

MTBirthPlace = MXBirthPlace

**ENDIF**

MTCountryandPlaceOfBirth = **Concatenate**(“5/”, MXBirthCountry, “/”, MTBirthPlace)

### 4.1.8 MX\_To\_MTPartyNameAndUnstructuredAddress

**Name**

MX\_To\_MTPartyNameAndUnstructuredAddress

**Name**

MX\_To\_MTPartyNameAndUntructuredAddress

**Business description**

The function translates an MX Party Name and PostalAddress.AddressLine to MT NameAndAddress format (4\*35)*.* If the AddressLine uses 3 lines and the Name is longer than 35 char, it will be truncated after 34 characters (and sign “+” added). If the AddressLine uses only 2 lines or less then the 70 first characters of Name will be copied on line 1 and line 2 (if needed). Truncation is possible. A sign”+” is added at the end of the data if truncation is needed.

IF MX AddressLine has value “NOTPROVIDED”, it will not be translated.

**Format**

**MX\_To\_MTPartyNameAndUnstructuredAddress**(MXParty ; MTNameAndAddress)

**Input**

MXParty: the entire structure of the MX party typed *PartyIdentification135*.

**Output**

MTNameAndAddress: name and address of the financial institution in the MT format (4\*35x).

**Preconditions**

None

**Formal description**

/\*Throughout the function, if translation of the source

component is spread over more than one line of the 4\*35x format, a Carriage Return Line Feed (*CRLF*) will be added

between consecutive lines to comply with the format of an MT field with multiple lines\*/

/\* Local variables

String: MXName,

MXNameAndAddress[]: table of string max 35 characters

Integer: NumberConsumedLines, AddressNumberOfLines \*/

AddressNumberOfLines = 0

MXName = MXParty.Name

**IF** **Length**(MXName)< 1 THEN

MXName = “NOTPROVIDED”

T12009

**ENDIF**

**IF** MXParty.PostalAddress.AddressLine **IsPresent AND**

MXParty.PostalAddress.AddressLine[1]**NOT EQUAL** to “NOTPROVIDED” THEN

AddressNumberOfLines = **NumberOfOccurrences**(AddressLine)

**ENDIF**

/\* Limit Name depending on Address Line \*/

**IF** AddressNumberOfLines = 3 THEN

/\* Name max 35 char \*/

**IF** **Length**(MXName)> 35 THEN

MXName = **Concatenate**(**Substring**(MXName, 1, 34), “+”)

**ENDIF**

**ELSE**

/\* Name max 70 char \*/

**IF** **Length**(MXName)> 70 THEN

MXName = **Concatenate**(**Substring**(MXName, 1, 69), “+”)

**ENDIF**

**ENDIF**

**IF** **Length**(MXName)< 36 THEN

MXNameAndAddress[1] = **Substring**(MXName, 1, 35)

NumberConsumedLines = 1

**ELSE**

/\* MXName is max 70 char \*/

MXNameAndAddress[1] = **Substring**(MXName, 1, 35)

MXNameAndAddress[2] = **Substring**(MXName, 36)

NumberConsumedLines = 2

**ENDIF**

**IF** MXParty.PostalAddress.AddressLine **IsPresent AND**

MXParty.PostalAddress.AddressLine[1]**NOT EQUAL** to “NOTPROVIDED” THEN

**For i = 1**,**NumberOfOccurrences**(MXParty.PostalAddress.AddressLine)

**IF Length**(MXParty.PostalAddress.AddressLine[i]) > 0 THEN

NumberConsumedLines = NumberConsumedLines +1

MXNameAndAddress[NumberConsumedLines]= MXParty.PostalAddress.AddressLine[i]

**ENDIF**

**END loop**

**ENDIF**

/\* Copy to MT structure \*/

**For i=1** to NumberConsumedLines

**AppendToNextLine**(MXNameAndAddress[i], MTNameAndAddress)

**END LOOP**

### 4.1.9 MX\_To\_MTPartyNameAndStructuredAddress

**Name**

MX\_To\_MTPartyNameAndStructuredAddress

**Business description**

This function translates a MX Name and structured Postal Address for a Party to a MT Structured address (subfield 2 in 50F or 59F).

The information will be translated following the order:

1/ Name

2/StreetName, BuildingNumber, BuildingName, Floor, PostBox, Room, Department, SubDepartment

3/Country/ TownName, PostCode, Country SubDivision, TownLocationName, DistrictName

Number 1/, 2/ and 3/ can be repeated but maximum 2 times for the same number.

If an element is missing there will be no indication it is missing in the MT string (ie., the position in the MT string will be insufficient to identify unambiguously the type of information)

The separator Comma will be used to separate the element in the MT string except between Country and TownName where “/” is used to be aligned with the MT structure for number “3/”

Country and TownName are mandatory in CBPR+ if structured postal address is used.

The presence of the LEI is taken into account to calculate the room left for the translation of the structured postal address. The LEI translation rules are defined in other functions. IF LEI is present then the Name is translated on 1 line. If LEI is absent, the Name can use 1 or 2 lines. Depending on the number of lines consumed by the LEI and Name, it may remain 2 to 3 lines.

Case where 3 lines are available and StreetName line is present then Country Line can use up to 2 lines if needed otherwise StreetName line can use up to 2 lines if needed. If Name length < 34, LEI is absent and StreetName line is absent, the Country line can then use up to 2 lines.

Case where 2 lines are available, StreetName line will use 1 line and Country line will use 1 line.

If StreetName line is absent, then Country line can use 2 lines.

After filling StreetName line and Country line, if there is one line with Number “1/” and Name is longer than 33 characters then one additional occurrence of “1/” is filled in.

The function handles the case where only Name is present.

**Format**

**MX\_To\_MTStructuredPartyNameAndStructuredAddress**(MXParty; MTNameAndAddress )

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

**Output**

MTNameAndAddress : max 4 lines of 35 char

**Preconditions**

None

**Formal description**

/\* Throughout the function, if translation of the source

component is spread over more than one line of the 4\*35x format, a Carriage Return Line Feed (*CRLF*) will be added

between consecutive lines to comply with the format of an MT field with multiple lines \*/

/\* Local variables

i, NextIndex, NumberOfNameOccurrences : integer

Separ : string

MXLEI, Temp : string

MXNumber1, MXNumber2, MXNumber3 :string

MXTable2[], MXTable3[], MTNameAndAddressTable[] : table of string

NumberRemainingLines : Integer

MXNumber2Present, NoStructuredAddress : Boolean \*/

/\* keep one line left for LEI if present. As the LEI is filled differently for Debtor and Creditor it will be filled outside the function \*/

MXTable2[1] = MXParty.PostalAddress.StreetName

MXTable2[2] = MXParty.PostalAddress.BuildingNumber

MXTable2[3] = MXParty.PostalAddress.BuildingName

MXTable2[4] = MXParty.PostalAddress.Floor

MXTable2[5] = MXParty.PostalAddress.PostBox

MXTable2[6] = MXParty.PostalAddress.Room

MXTable2[7] = MXParty.PostalAddress.Department

MXTable2[8] = MXParty.PostalAddress.SubDepartment

MXTable3[1] = MXParty.PostalAddress.Country

MXTable3[2] = MXParty.PostalAddress.TownName

MXTable3[3] = MXParty.PostalAddress.PostCode

MXTable3[4] = MXParty.PostalAddress.CountrySubdivision

MXTable3[5] = MXParty.PostalAddress.TownNameLocation

MXTable3[6] = MXParty.PostalAddress.DistrictName

MXLEI = MXParty.Identification.OrganisationIdentifcation.LEI

Separ = “,”

MXNumber1 = MXParty.Name

**IF** **Length**(MXNumber1) = 0 THEN

MXNumber1 = “NOTPROVIDED”

~~T20130~~ T12009

/\* This is not expected in pacs because if BIC is absent then Name is mandatory in CBPR+ but possible in camt (eg., camt.054) \*/

**ENDIF**

/\* IF LEI is present, use 1 line for Name else use 2 lines \*/

**IF** **Length**(MXLEI)> 0 THEN

**IF** **Length**(MXNumber1)> 33 THEN

MXNumber1 = **Concatenate**(**Substring**(MXNumber1), 1, 32), “+”)

**ENDIF**

MTNameAndAddressTable[1]= **Concatenate**(“1/”, MXNumber1)

/\* 1 line will be used by LEI in another calling function, 1 by the Name \*/

NumberRemainingLines = 2

**ELSE**

/\* 2 lines can be used for the name if needed \*/

**IF** **Length**(MXNumber1)> 66 THEN

MTNameAndAddressTable[1]= **Concatenate**(“1/”,**Substring**(MXNumber1,1,33))

MTNameAndAddressTable[2]= **Concatenate**(“1/”,**Substring**(MXNumber1,34,32), “+”)

NumberRemainingLines = 2

**ELSEIF** **Length**(MXNumber1)> 33 THEN

MTNameAndAddressTable[1]=

**Concatenate**(“1/”,**Substring**(MXNumber1,1,33))

MTNameAndAddressTable[2]=

Concatenate(“1/”,**Substring**(MXNumber1,34))

NumberRemainingLines = 2

**ELSE**

MTNameAndAddressTable[1]=

**Concatenate**(“1/”,MXNumber1)

NumberRemainingLines = 3

**ENDIF**

**ENDIF** /\*LENGTH(MXLEI)\*/

**IF** MXParty.PostalAddress.Country **NOT IsPresent** THEN

/\* No structured Postal address. Copy the Name to MTNameandAddress \*/

**For i** = 1 to (4 – NumberRemainingLines)

**IF** **Length**(MTNameAndAddressTable[i])> 0 THEN

**AppendToNextLine**(MTNameAndAddressTable[i], MTNameAndAddress)

**ELSE**

EXIT loop

**ENDIF**

**END loop**

Exit function

**ENDIF** /\* country not present \*/

/\* Fill in MXNumber2 respecting the following order StreetName, BuildingNumber, BuildingName, Floor, PostBox, Room, Department, SubDepartment and using the separator between elements. Missing information is not indicated. \*/

**For i = 1 to 8**

{

**IF** **Length**(MXTable2[i])> 0 THEN

**IF** **Length**(MXNumber2) = 0 THEN

/\* it is the first element found \*/

MXNumber2 = **Concatenate**(MXTable2[i])

**ELSE**

MXNumber2 = **Concatenate**(MXNumber2,Separ, MXTable2[i])

**ENDIF**

**ENDIF**

}

**Next i**

/\* Fill in MXNumber3 respecting the following order Country, TownName, PostCode, CountrySubDivision, TownLocationName, DistrictName and using the separator between elements except between Country and TownName where “/” is used. In CBPR+, if Structured postal address is used, Country and TownName are mandatory \*/

/\* Start with Country and TownName \*/

MXNumber3 = **Concatenate**(MXTable3[1], “/”)

MXNumber3 = **Concatenate**(MXNumber3, MXTable3[2])

/\* Continue with other elements \*/

**For i** = 3 to 6

{**IF** **Length**(MXTable3[i])> 0 THEN

MXNumber3 = Concatenate (MXNumber3,Separ, MXTable3[i])

**ENDIF**

}

**Next i**

/\* Check if MXNumber2 has meaningful information \*/

**IF** **Length**(MXNumber2) > 0 THEN

MXNumber2Present = “true”

**ELSE**

MXNumber2Present = “false”

**ENDIF**

**Case 1**

**IF** NumberRemainingLines = 3 THEN

NextIndex = 4 – NumberRemainingLines + 1

**IF** MXNumber2Present THEN

/\* MXNumber3 can use 2 lines, if needed, else MXNumber2 can use 2 lines if needed \*/

**Call** Subfunction **Case11** /\* described below \*/

**ELSE**

/\* MXNumber2 is absent, 2 lines can be used by MXNumber3 as 3/ is limited to 2 occurrences as per MT UHB 2021\*/

**Call** Subfunction **Case12** /\* described below \*/

**ENDIF**

**ENDIF**

**Case 2**

**IF** NumberRemainingLines = 2 THEN

NextIndex = 4 – NumberRemainingLines + 1

**IF** MXNumber2Present THEN

/\* MXNumber2 uses 1 line and MXNumber3 uses 1 line \*/

**Call** Subfunction **Case21**

**ELSE**

/\* MXNumber3 can use 2 lines \*/

**Call** Subfunction **Case22**

**ENDIF**

**ENDIF**

/\* **Optimise** and add a line for Name if Name has been limited to 33 char. This could happen if MXNumber2 is absent and only 1 line is used by MXNumber3. Then the Name has been too much restricted but max 2 occurrences of “1/” is allowed in MT\*/

**For i** = 1 to 4

**IF Substring**(MTNameAndAddressTable[i], 1, 2) = “1/” THEN

NumberOfNameOccurrences = NumberOfNameOccurrences + 1

**ENDIF**

**Next i**

/\* Recalculate the number of Remaining Lines, eg., number of Lines empty \*/

NumberRemainingLines = 4

**For i** = 1 to 4

**IF Length**(MTNameAndAddressTable[i]) > 0 THEN

NumberRemainingLines = NumberRemainingLines – 1

**ELSE**

Exit loop

**ENDIF**

**Next i**

**IF** NumberRemainingLines = 2 AND NumberOfNameOccurrences = 1 AND **Length**(MXParty.Name) > 33 THEN

/\* Add a line with Name information. NumberOfNameOccurrences = 1 means either that Name does not require more space or LEI is present and Name has been limited. IF LEI is present then it must remain 2 lines out of which one will be consumed by LEI in another function. \*/

MTNameAndAddressTable[3]= MTNameAndAddressTable[2]

**IF** **Length**(MXNumber1)> 66 THEN

MTNameAndAddressTable[1]= **Concatenate**(“1/”,**Substring**(MXNumber1,1,33))

MTNameAndAddressTable[2]= **Concatenate**(“1/”,**Substring**(MXNumber1,34,32), “+”)

**ELSEIF** **Length**(MXNumber1)> 33 THEN

MTNameAndAddressTable[1]=

**Concatenate**(“1/”,**Substring**(MXNumber1,1,33))

MTNameAndAddressTable[2]=

**Concatenate**(“1/”,**Substring**(MXNumber1,34))

**ENDIF**

**ENDIF**

/\* Fill in the MT structure with CRLF between the lines \*/

**For i** = 1 to 4

**IF Length**(MTNameAndAddressTable[i])> 0 THEN

**AppendToNextLine**(MTNameAndAddressTable[i], MTNameAndAddress)

**ELSE**

EXIT loop

**ENDIF**

**Next i**

**Subfunction definitions**

**Subfunction Case11**

/\* MXNumber3 can use 2 lines, if needed, else MXNumber2 can use 2 lines if needed \*/

**IF** **Length**(MXNumber3)> 66 THEN

/\* MXNumber3 uses 2 lines, MXNumber2 uses 1 line \*/

/\* Fill in with MXNumber2 \*/

**IF** **Length**(MXNumber2) > 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”, **Substring**(MXNumber2,1,32), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]=**Concatenate**(“2/”, MXNumber2)

**ENDIF**

NextIndex = NextIndex + 1

/\* Fill in 2 lines with MXNumber3\*/

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,34,32), “+”)

**ELSEIF Length**(MXNumber3)> 33 THEN

/\* MXNumber 3 uses 2 lines but no truncation needed for MXNumber3. MXNumber2 uses 1 line \*/

/\* Fill in with MXNumber2 \*/

**IF** **Length**(MXNumber2) > 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”, **Substring**(MXNumber2,1,32), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”, MXNumber2)

**ENDIF**

NextIndex = NextIndex + 1

/\* Fill in 2 lines with MXNumber3\*/

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,34))

**ELSE**

/\* LENGTH(MXNumber3)<= 33 uses 1 line , MXNumber2 can use 2 lines if needed \*/

/\* Fill in with MXNumber2 \*/

**IF** **Length**(MXNumber2) > 66 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”,**Substring**(MXNumber2, 1, 33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”,**Substring**(MXNumber2, 34, 32), “+”)

NextIndex = NextIndex + 1

**ELSE IF** **Length**(MXNumber2) > 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”,**Substring**(MXNumber2, 1, 33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”,**Substring**(MXNumber2, 34))

NextIndex = NextIndex + 1

**ELSE** /\* MXNumber2 uses 1 line \*/

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”,MXNumber2)

NextIndex = NextIndex + 1

**ENDIF**

MTNameAndAddressTable[NextIndex]= Concatenate(“3/”,MXNumber3)

**ENDIF** /\* ENDIF Length(MXNumber3)> 66 \*/

**Subfunction** **Case12**

/\* MXNumber2 is absent, 2 lines can be used by MXNumber3 as per SR2021 MT UHB \*/

**IF** **Length**(MXNumber3)> 66 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,34,32), “+”)

**ELSEIF** **Length**(MXNumber3)> 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,34))

**ELSE**

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, MXNumber3)

**ENDIF**

**Subfunction** **Case21**

/\* MXNumber2 uses 1 line and MXNumber3 uses 1 line \*/

**IF** **Length**(MXNumber2) > 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”, **Substring**(MXNumber2,1,32), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]= **Concatenate**(“2/”,MXNumber2)

**ENDIF**

NextIndex = NextIndex + 1

**IF** **Length**(MXNumber3) > 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,32), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”,MXNumber3)

**ENDIF**

**Subfunction** **Case22**

/\* MXNumber3 can use 2 lines \*/

**IF** **Length**(MXNumber3) > 66 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,34,32), “+”)

**ELSEIF** **Length**(MXNumber3) > 33 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,1,33))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, **Substring**(MXNumber3,34))

**ELSE**

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, MXNumber3)

**ENDIF**

### 4.1.10 MX\_To\_MTPartyNameAndAddressLEI2

**Name**

MX\_To\_MTPartyNameAndAddressLEI2

**Business description**

The function is used to create subfield2 in 59F when MXAddressLine is structured and contains lines starting with “2/” or “3/” coming from previous MT 59F or original MX with structured postal address translated to MT. If there is still room left once the MXAddressLine is translated, Name is translated (minimum 33 characters). If there is still room left and LEI is present and the number of occurrences of “3/” in the MXAddressLine is limited to 1, LEI is translated. Otherwise, if there is still room left, Name can use up to 2 lines with number “1/”.

Lines with number “1/”, “2/” and “3/” can be repeated but maximum twice for a same Number.

LEI structure is “3/CountryCode/LEIC/LEIIdentifier”.

**Format**

**MX\_To\_MTPartyNameAndAddressLEI2**(MXParty; MTNameAndAddress)

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

**Output**

MTNameAndAddress with the format defined in 59F subfield 2

**Preconditions**

MX Address Line is present and structured with Number “2/” and “3/”

**Formal description**

/\* Local variables

MTNameAndAddressTable [ ] : table of string

AddressLineNumberOfOccurrences, index, i , Number3Occurrences : integer

MXName : string

MXLEI, MXCountryCode : string

\*/

Number3Occurrences = 0

**IF** **Length**(MXParty.Name) = 0 THEN

MXName = “NOTPROVIDED”

**ELSE**

MXName = MXParty.Name

**ENDIF**

AddressLineNumberOfOccurrences = **NumberOfOccurrences**(MXParty.PostalAddress.AddressLine)

/\* count the number of lines in MX AddressLine starting with “3/” \*/

**For** i = 1 to AddressLineNumberOfOccurrences

**IF** **Substring**(MXParty.PostalAddress.AddressLine[i],1,2) = “3/” THEN

Number3Occurrences = Number3Occurrences + 1

**ENDIF**

**Next i**

/\* LEI can be translated only if there is maximum 1 line starting with “3/” in the MX AddressLine \*/

**IF Length**(MXParty.Identification.OrganisationIdentification.LEI) > 0 THEN

**IF** Number3Occurrences < 2 THEN

**{** MXLEI = MXParty.Identification.OrganisationIdentification.LEI

/\* Search for the country code \*/

**For index** = 1 to AddressLineNumberOfOccurrences

**IF** **IsPresentPattern**(**Substring**(MXParty.PostalAddress.AddressLine[index],1,2), “3/”)

THEN

/\* country code is expected in the first “3/” occurrence \*/

**IF IsCountry**(**Substring**(MXParty.PostalAddress.AddressLine[index], 3,2) THEN

MXCountryCode = Substring(MXParty.PostalAddress.AddressLine[index], 3,2)

Exit loop

**ENDIF**

**ENDIF**

**Next index**

**END loop**

/\* Fill in with 3/CountryCode/LEIC/LEIIdentifier \*/

IF LENGTH(MXCountryCode) = 0 THEN

MXCountryCode = “CH”

ENDIF

MTLEI = Concatenate(“3/”,MXCountryCode,”/LEIC/”, MXLEI)**}**

**ELSE**

/\* Number3Occurrences >= 2, LEI is not translated \*/

Flag\_MissingInformation = “True” **ENDIF** /\* End Number3Occurrences < 2 \*/**ENDIF** /\* Length LEI \*/

**IF** AddressLineNumberOfOccurrences = 3 THEN

/\* Name length is limited to 1 line \*/

**IF Length**(MXName) > 33 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,32),”+”)

**ELSE**

MTNameAndAddressTable [1] = Concatenate(“1/”, MXName)

**ENDIF**

**IF Length**(MXLEI) > 0 THEN

Flag\_MissingInformation = “True”

**ENDIF**

index = 1

**For i** = 1 to AddressLineNumberOfOccurrences

index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

**ELSEIF** AddressLineNumberOfOccurrences = 2 THEN

/\* IF LEI is present and number of lines starting with “3/” in MX AddressLine is less that 2 (see above), then Name is limited to 1 line else Name can use 2 lines \*/

**IF** **Length** (MXLEI) > 0THEN

**IF** **Length**(MXName) > 33 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,32),”+”)

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

**ENDIF**

index = 1

**For i** = 1 to AddressLineNumberofOccurrences

index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

index = index + 1

MTNameAndAddressTable [index] = MTLEI

**ELSE** /\* AddressLineNumberOfOccurrences = 2, no LEI or no translation of LEI because there is already 2 lines starting with “3/” in MX Address Line, Name can use up to 2 lines \*/

**IF Length**(MXName) > 66 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, **Substring**(MXName, 34,32), “+”)

Index = 2

**ELSEIF Length**(MXName) > 33

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, Substring(MXName,34))

Index = 2

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

Index = 1

**ENDIF**

**For i** = 1 to AddressLineNumberofOccurrences

index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

**ELSEIF** AddressLineNumberOfOccurrences = 1

/\* meaning no line 2/ only 3/ in AddressLine\*/

/\* Name can use up to 2 lines \*/

/\* 1 line can be used by LEI if present \*/

**IF Length**(MXName) > 66 THEN

MTNameAndAddressTable [1] = **Concatenate**(“1/”, **Substring**(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, **Substring**(MXName, 34,32), “+”)

index = 2

**ELSEIF** **Length**(MXName) > 33

MTNameAndAddressTable [1] = **Concatenate**(“1/”, Substring(MXName,1,33))

MTNameAndAddressTable [2] = **Concatenate**(“1/”, Substring(MXName,34)))

index = 2

**ELSE**

MTNameAndAddressTable [1] = **Concatenate**(“1/”, MXName)

index = 1

**ENDIF**

**For i** = 1 to AddressLineNumberofOccurrences

index = index + 1

MTNameAndAddressTable [index] = MXParty.PostalAddress.AddressLine[i]

**Next i**

/\* IF LEI is present and number of occurrences starting with “3/” in MX AddressLine is less than 2 \*/

**IF Length**(MXLEI) > 0 THEN

index = index + 1

MTNameAndAddressTable [index] = MTLEI

**ENDIF**

**ENDIF**  /\* IF AddressLineNumberOfOccurrences = 3 \*/

/\* Fill in the MTNameAndAddress structure \*/

**For i= 1** to index

**AppendToNextLine**(MTNameAndAddressTable[i], MTNameAndAddress)

**Next i**

### 4.1.11 MX\_To\_MTPartyNameAndAddressLEI1

**Name**

MX\_To\_MTPartyNameAndAddressLEI1

**Business description**

The function is used to create subfield2 in 59F when MX Structured Postal Address is present.. One line is reserved for LEI, if present in the function MX\_To\_MTPartyNameAndStructuredAddress(1). The functions MX\_To\_MTPartyNameAndStructuredAddress(1) define the translation priority for the elements from the structured postal address. LEI structure is “3/CountryCode/LEIC/LEIIdentifier.” The numbers “1/”, “2/” and “3/” can be repeated but only twice for the same number, the presence of LEI limits to 1 line the information from CountryLine starting with “3/”.

**Format**

**MX\_To\_MTPartyNameAndAddressLEI1**(MXParty; MTNameAndAddress)

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

**Output**

MTNameAndAddress with the format defined in 59F subfield 2

**Preconditions**

None

**Formal description**

/\* Local variables

MXCountryCode, MTLEI : string \*/

**IF Length**(MXParty.Identification.OrganisationIdentification.LEI) > 0

THEN

MXLEI = MXParty.Identification.OrganisationIdentification.LEI

**IF** MXParty.PostalAddress.Country **IsPresent** THEN

MXCountryCode = MXParty.PostalAddress.Country

**ELSE**

/\* should not be needed as country is mandatory in structured postal address \*/

MXCountryCode = “CH”

**ENDIF**

/ Fill in with 3/CountryCode/LEIC/LEIIdentifier

MTLEI = **Concatenate**(“3/”,MXCountryCode,”/LEIC/”, MXLEI)

**ENDIF**

/\* Fill in 1/, 2/ and 3/ \*/

**IF Length**(MXLEI)> 0 THEN

/\* As one line is used for 1/ and one line for 3/LEI, then the MXAddress can only generate 1 line 3/ and 1 line 2/ \*/

**MX\_To\_MTPartyNameAndStructuredAddress1**(MXParty; MTNameAndAddress)

**ELSE**

**MX\_To\_MTPartyNameAndStructuredAddress**(MXParty; MTNameAndAddress)

**ENDIF**

/\* Check if still room left. Expected one line left if LEI is present \*/

**IF ReturnFirstLineEmpty** (MTNameAndAddress, 4) = 0 AND **Length**(MXLEI) > 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function

**ENDIF**

**IF** **Length**(MXLEI) > 0 THEN

/\* Append LEI \*/

**AppendToNextLine**(MTLEI, MTNameAndAddress)

**ENDIF**

### 4.1.12 MX\_To\_MTPartyNameAndStructuredAddressNoNumber

**Name**

MX\_To\_MTPartyNameAndStructuredAddressNoNumber

**Business description**

This function translates a MX Name and structured Postal Address for a Party to a MT Name and Address without Number (ie., without n/ as in 50F or 59F). The logic is similar to the one applied in MX\_To\_MTFinancialInstitutionNameAndStructuredAddress but the MX element type and path are different.

**Format**

**MX\_To\_MTFPartyNameAndStructuredAddressNoNumber**(MXParty; MTNameAndAddress )

**Input**

MXParty: the entire structure of the MXAgent typed PartyIdentification135.

**Output**

**MTNameAndAddress : max 4 lines of 35 char**

**Formal description**

In order to reuseMX\_To\_MTFinancialInstitutionNameAndStructuredAddress, variables in that function must be initiated differently.

MXPostalAddress = MXParty.PostalAddress

MXNumber1 = MXParty.Name

**Then apply the same logic as described in** MX\_To\_MTFinancialInstitutionNameAndStructuredAddress

## 4.2 Financial Institution Translation Rule

The translation rule descriptions provided in this section are for translation rules that relate to Financial Institution information.

### 4.2.1 MX\_To\_MTBICFI

**Name**

MX\_To\_MTBIC FI

**Business description**

The function translates an MX BICFI to an MT BICFI. The MX BICFI is found in a component typed *BranchAndFinancialInstitutionIdentification6*. All other identifications that are allowed in the structure, ClearingSystemMemberIdentification, Name, PostalAddress and Other, are truncated by this function.

**Format**

**MX\_To\_MTBICFI**(MXAgentIdentification ; MTBIC)

**Input**

MXAgentIdentification: the entire structure of the MX agent identification typed *BranchAndFinancialInstitutionIdentification6*.

**Output**

MTBICFI: BIC of the financial institution in the MT format.

**Preconditions**

None.

**Formal description**

MTBIC = FinancialInstitutionIdentification.BICFI

**Example 1 MX Source:**

MXIntermediaryAgent1

<IntrmyAgt1>

<FinInstnId>

<BICFI>COBADEFF</BICFI>

</FinInstnId>

</IntrmyAgt1>

**MT Translation:**

:56A:COBADEFF

### 4.2.2 MX\_To\_MTClearingIdentifier

**Name**

MX\_To\_MTClearingIdentifier

**Business description**

The function translates an MX ClearingSystemMemberIdentification to an MT clearing identifier. The clearing system identification and the MemberIdentification are mandatory in the ClearingSystemMemberIdentification element. In CBPR+, only ISO codes are allowed for the Clearing System Identification.

It is checked whether this 5 characters code is present in the ISO externalized MX ClearingSystemList (Annex to this function). If the code is found, the equivalent MT code will be used in combination with the MemberIdentification element to obtain the translated MT clearing identifier. If the code is not found in the list, then the content of the identification element “as is” will be taken in combination with the MemberIdentification element to obtain the translated MT clearing identifier.

Finally a double slash “//” prefix is added to the translated MX clearing identification in line

with the MT formatting rules.

**Format**

**MX\_To\_MTClearingIdentifier**(MXAgentIdentification ; MTPartyIdentifier)

**Input**

MXAgentIdentification: the entire structure of the MX agent identification typed *BranchAndFinancialInstitutionIdentification6*.

**Output**

MTPartyIdentifier: clearing system member identification of the financial institution in the MT format (/34x).

**Preconditions**

None

**Formal description**

/\*Extract ClearingSystemMemberIdentification from

MXAgentIdentification. MXClearingSystem, MXClearingMemberCode are local variables\*/

/\*For reason of function readability, the

“FinancialInstitutionIdentification.ClearingSystemMemberIdentification” path is replaced by “*F.Cl*””\*/

**IF** **IsPresent**(*F.Cl.*ClearingSystemIdentification.Code) THEN

MXClearingSystem = *F.Cl.*ClearingSystemIdentification.Code

MXClearingMemberCode= *F.Cl.*MemberIdentification

/\*Check whether the MXClearingSystem belongs to the list\*/ **IF WithinList**(MXClearingSystem, MXClearingSystemList) THEN

/\*Get the MTClearingSystem in the list equivalent to the MXClearingSystem. MTClearingSystem is a local variable\*/

MTClearingSystem = **EquivalentCode**(MXClearingSystem,

MXClearingSystemList, MTClearingSystemList)

**ELSE**

/\*MXClearingSystem kept as is\*/

/\* Example SGIBG, THCBC, TWNCC, SESBA \*/

MTClearingSystem = MXClearingSystem

**ENDIF**

/\*Build output parameter. MTClearingId is a local variable\*/

MTClearingId = **Concatenate**(“//", MTClearingSystem, MXClearingMemberCode)

/\*Build output string\*/

/\*Special check in case a BIC is present with a Fedwire clearing identifier. As per SWIFT User Handbook, Message Reference Guides for Category 1 and Category 2, a Fedwire

Routing Number in combination with a BIC (option A) must be used without the 9-digit code. Only “//FW” is allowed serving as a clearing channel. But this function does not translate the clearing channel “RTGS” on the agents (see rather MX\_To\_MT\_ClearingIdentifierAndChannel). So it is not correct to replace “//FWnnn..” by “//FW”. Therefore the ClearingSystemMemberID is not translated in case of option A \*/

/\*Check whether a BIC is present. As per SWIFT User Handbook, Message Reference Guides for Category 1 and for Category 2, for some clearing systems the use of a clearing identification in combination with a BIC (MT option A) is not allowed: CHIPS Participant Identifier (“CP”), Russian Central Bank

Identification Code (“RU”), Swiss Clearing Code (“SW”)\*/ /\*System code is CP, RU or SW and BIC is present\*/

/\* Assumption is made that these clearing systems will not occur in 53a, 54a and 55a for which there is no restrictions in the clearingSystemIdentification allowed. If this is the case, then BIC is sufficient to uniquely identify the Agent \*/

**IF** **Substring**(MTClearingId, 3, 2) = “FW” AND

**IsPresent**(FinancialInstitutionIdentification.BIC) THEN

MTPartyIdentifier = “”

**T11005**

**ELSEIF**

**Substring**(MTClearingId, 3, 2) = “CP” OR “RU” OR “SW” AND

**IsPresent**(FinancialInstitutionIdentification.BIC) THEN

MTPartyIdentifier = “”

**T11006**

/\*In all other cases the translated MX clearing Member identification is copied in the output parameter\*/

**ELSE**

MTPartyIdentifier = MTClearingId

**ENDIF**

**ENDIF**  /\* IF IsPresent(*F.Cl.*ClearingSystemIdentification.Code)\*/

**For MT and ISO 20022 “Externalised” MX Clearing System Lists, please refer to section 5 Annex.**

**Example 1 MX Source:**

A ClearingSystemMemberIdentification of a DebtorAgent

<DbtrAgt>

<FinInstnId>

<BIC>COBADEFF</BIC>

<ClrSysMmbId>

<ClrSysId>

<Cd>DEBLZ</Cd>

</ClrSysId>

<MmbId>12345678</MmbId>

</ClrSysMmbId>

</FinInstnId>

</DbtrAgt>

**MT Translation:**

:52A://BL12345678

*COBADEFF*

*(Italic not translated by this function)*

**Example 2 MX Source:**

A ClearingSystemMemberIdentification of a CreditorAgent

<CdtrAgt>

<FinInstnId>

<ClrSysMmbId>

<ClrSysId>

<Cd>AUBSB</Cd>

</ClrSysId>

<MmbId>1234572</MmbId>

</ClrSysMmbId>

*<Nm>BANK ABC</Nm>*

*<PstlAdr>*

*<TwnNm>UTOPIA</TwnNm>*

*<Ctry>BE</Ctry>*

*</PstlAdr>*

</FinInstnId>

</CdtrAgt>

**MT Translation:**

:57D://AU1234572

*BANK ABC*

*BE/UTOPIA*

*(Italic not translated by this function)*

**Example 3**

**MX Source:**

A ClearingSystemMemberIdentification of a CreditorAgent <CdtrAgt>

<FinInstnId>

*<*ClrSysMmbId>

<*BIC>IRVTUS3N</BIC>*

<ClrSysId>

<Cd>USABA</Cd>

</ClrSysId>

<MmbId>123456789</MmbId>

</ClrSysMmbId>

</FinInstnId>

</CdtrAgt>

**MT Translation:**

:57A://FW

*IRVTUS3N*

*(* *Italic not translated by this function)*

### 4.2.3 MX\_To\_MTClearingIdentifierAndChannel

**Name**

MX\_To\_MTClearingIdentifierAndChannel

**Business description**

The function translates an MX clearing channel indicator “RTGS” to an MT clearing channel indicator “//RT” or “//FW”. In case the MX agent identification contains a clearing identification, it will be translated to an MT clearing identification. Depending on whether the MX agent identification contains a BIC and on whether combining a BIC with a clearing identification is allowed for the clearing system (as per *SWIFT User Handbook, Message Reference Guides for Category 1 and Category 2*), the clearing channel indicator will or will not be combined with the clearing identification in the output string. If the combination would bypass the maximum format of the MTPartyIdentifier translation target (/34x), then only the clearing identification will be written and the clearing channel indicator dropped.

**Format**

**MX\_To\_MTClearingIdentifierAndChannel**(MXAgentIdentification ; MTPartyIdentifier)

**Input**

MXAgentIdentification: the entire structure of the MX agent identification typed *BranchAndFinancialInstitutionIdentification6*.

**Output**

MTPartyIdentifier: clearing channel and possible clearing code identifier of the financial institution in the MT format (/34x).

**Preconditions**

PaymentTypeInformation/ClearingChannel = “RTGS”

**Formal description**

/\*Translation ClearingSystemMemberIdentification if present by calling a sub-function. MTClearingId is a local variable\*/

/\*For reason of function readability, the

“FinancialInstitutionIdentification.ClearingSystemMemberIdentification”

path is replaced by “*F.Cl*””\*/

**IF** **IsPresent**(*F.Cl.*ClearingSystemIdentification.Code) THEN

MXClearingSystem = *F.Cl.*ClearingSystemIdentification.Code

MXClearingMemberCode = *F.Cl.*MemberIdentification

/\*Check whether the MXClearingSystem belongs to the list\*/

**IF** **WithinList**(MXClearingSystem, MXClearingSystemList)THEN

/\*Get the MTClearingSystem in the list equivalent to the MXClearingSystem. MTClearingSystem is a local variable\*/

MTClearingSystem = **EquivalentCode**(MXClearingSystem,

MXClearingSystemList, MTClearingSystemList)

**ELSE**

/\*MXClearingSystem kept as is\*/

/\* Example SGIBG, THCBC, TWNCC, SESBA \*/

MTClearingSystem = MXClearingSystem

**ENDIF**

/\* Build output parameter \*/

MTClearingId = **Concatenate**(“//”, MTClearingSystem, MXClearingMemberCode)

**ELSE**

/\*Assign emptystring\*/

MTClearingId = “”

**ENDIF**

/\*Build output string\*/

/\*Special check in case a BIC is present with country code

“US”. As per User Handbook documentation, a Fedwire Routing

Number in combination with a BIC (option A) must be used without the 9-digit code. Only “//FW” can be indicated serving as a clearing channel\*/

**IF** **IsPresent**(FinancialInstitutionIdentification.BIC) AND

**Substring**(FinancialInstitutionIdentification.BIC, 5, 2) = “US” THEN

MTPartyIdentifier = “//FW”

**IF** **IsPresent**(*F.Cl.*ClearingSystemIdentification.Code)THEN

**T11005**

**ENDIF**

**IF** **IsPresent**(*F.Cl.*ClearingSystemIdentification.Code)THEN

**T11007**

**ENDIF**

/\*No BIC is present, but clearing system code is “FW”, not to be combined with “//RT”\*/

**ELSEIF Substring**(MTClearingId, 3, 2) = “FW” THEN

MTPartyIdentifier = MTClearingId

/\*In all other cases “//RT” is the default translated clearing channel indicator, possibly combined with a clearing identification if there is no BIC \*/

**ELSE**

/\*Check whether the ClearingChannel indicator “//RT” (4 characters) combined with the translated clearing identification MTClearingId fits the MTPartyIdentifier format (/34x), meaning that the length of the MTClearingId must be less than 32 characters\*/

**IF** **Length**(MTClearingId) < 32 THEN

MTPartyIdentifier = **Concatenate**(“//RT”, MTClearingId)

**ELSE**

MTPartyIdentifier = MTClearingId

**ENDIF**

**ENDIF**

**Example 1 MX Source:**

Clearingchannel “RTGS” in combination with an MX

IntermediaryAgent1

<IntrmyAgt1>

<FinInstnId>

*<BIC>GKCCBEBB</BIC>*

</FinInstnId>

</IntrmyAgt1>

**MT Translation:**

:56A://RT

*GKCCBEBB*

*(Italic not translated by this function)*

**Example 2 MX Source:**

Clearingchannel “RTGS” in combination with an MX CreditorAgent

<CdtrAgt>

<FinInstnId>

<ClrSysMmbId>

<ClrSysId>

<Cd>GBDSC</Cd>

</ClrSysId>

<MmbId>123456</MmbId>

</ClrSysMmbId>

*<Nm>Bank ABC</Nm>*

*<PstlAdr>*

*<TwnNm>London</TwnNm>*

*<Ctry>GB</Ctry> </PstlAdr>*

</FinInstnId>

</CdtrAgt>

**MT Translation:**

:57D://RT//SC123456

*Bank ABC*

*GB/London*

*(Italic not translated by this function)*

**Example 3 MX Source:**

Clearingchannel “RTGS” in combination with an MX

IntermediaryAgent1

<IntrmyAgt1>

<FinInstnId>

*<BIC>IRVTUS3N</BIC>*

</FinInstnId>

</IntrmyAgt1>

**MT Translation:** :56A://FW

*IRVTUS3N*

*(Italic not translated by this function)*

**Example 4 MX Source:**

Clearingchannel “RTGS” in combination with an MX

IntermediaryAgent1

<IntrmyAgt1>

<FinInstnId>

<ClrSysMmbId>

<ClrSysId>

<Cd>USABA</Cd>

</ClrSysId>

<MmbId>123456789</MmbId>

</ClrSysMmbId>

*<Nm>Bank ABC</Nm>*

*<PstlAdr>*

*<TwnNm>New York</TwnNm>*

*<Ctry>US</Ctry>*

*</PstlAdr>*

</FinInstnId>

</IntrmyAgt1>

**MT Translation:**

:56D://FW123456789

*Bank ABC*

*US/New York*

*(Italic not translated by this function)*

### 4.2.4 MX\_To\_MTClearingIdentifierChoice

**Name**

MX\_To\_MTClearingIdentifierChoice

**Business description**

The function calls MX\_To\_MTClearingIdentifierAndChannel if ClearingChannel is present with value “RTGS” otherwise the function MX\_To\_MTClearingIdentifier is called.

**Format**

**MX\_To\_MTClearingIdentifierChoice**(MXAgentIdentification, MXClearingChannel ; MTPartyIdentifier)

**Input**

MXAgentIdentification: the entire structure of the MX agent identification typed *BranchAndFinancialInstitutionIdentification6*.

MXClearingChannel typed ClearingChannel2Code (from CreditTransferTransactionInformation/PaymentTypeInformation/ClearingChannel)

**Output**

MTPartyIdentifier: clearing channel and possible clearing code identifier of the financial institution in the MT format (/34x).

**Preconditions**

None

**Formal description**

**IF** MXClearingChannel= “RTGS” THEN

MTPartyIdentifier = MX\_To\_MTClearingIdentifierAndChannel

**ELSE**

MTPartyIdentifier = MX\_To\_MTClearingIdentifier

**ENDIF**

### 4.2.5 MX\_To\_MTNameAndAddressToClearingSystemIdentifier

**Name**

MX\_To\_MTNameAndAddressToClearingSystemIdentifier

**Business description**

The function is called when the original message is an MT message with a ClearingSystemIdentifier which has no equivalent in the ISO list and when there is no BIC and no Name and Address to identify the Financial Institution. In that case the ClearingSystemIdentifier has been translated from MT to MX in the Party Name and AddressLine keeping the “//” in order to identify it is a ClearingSystemMemberIdentification. This function is translating back to MT to a format B or C.

**Format**

**MX\_To\_MTNameAndAddressToClearingSystemIdentifier**(MXAgentIdentification ; MTPartyIdentifier)

**Input**

MXAgentIdentification: the entire structure of the MX agent identification typed *BranchAndFinancialInstitutionIdentification6*.

**Output**

MTPartyIdentifier: clearing system code of the financial institution in the MT format (/34x).

**Preconditions**

Used to fill a format B or C only.

MXAgentIdentification BIC is absent and MXAgentIdentification ClearingSystem MemberIdentifier is absent, MXAgentIdentification Name length <= 35 and MXAgentIdentification Name = MXAgentIdentification addressLine [1]; MXAgentIdentification Name starts with “//”.

**Formal description**

/\* Extract the Name to get the MTClearingCode format \*/

/\* Local variables:

MXName : string \*/

MTPartyIdentifier = MXPartyIdentifier.FinancialInstitutionIdentification.Name

### 4.2.6 MX\_To\_MTAgent

**Name**

MX\_To\_MTAgent(MXAgent, MXCodeLength; MTAgent)

**Business description**

The function extracts the BICFI or the Name and Postal Address (Country,TownName) or the ClearingSystemMemberIdentification depending on presence of the elements. Name and Postal address are concatenated according to the following pattern (Name, “(“,Country,”(“, TownName)

IF Name is translated, Name length is calculated in order to fill max 2 lines in MT (including the MX code word) Preference is given to BICFI.

The ISO Clearing System Identification is not converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

**Format**

**Input**  MXAgent : BranchAndFinancialInstitutionIdentification6

MXCodeLength : length of the codeword to be used to translate to MT including the starting and ending slashes (example /INTA/)

**Output** MTAgent : string

**Preconditions**

None.

**Formal description**

/\* Local variables : MXBICFI, MXName, MXTownName, MXCountry, MXClearingId:string

MaxCCNameLength,MaxNameLength : integer \*/

MXBICFI = MXAgent.FinancialInstitutionIdentification.BICFI

MXName = MXAgent.FinancialInstitutionIdentification.Name

MXTownName = MXAgent.FinancialInstitutionIdentification.PostalAddress.TownName

MXCountry = MXAgent.FinancialInstitutionIdentification.PostalAddress.Country

MXClearingId = MXAgent.FinancialInstitutionIdentification.ClearingSystemMemberIdentification

/\* IF Name is translated and IF Country is present then the structure is

“/CodeWord/Name(Country(TownName” that can use up to max 2 lines of 35 char, first line starting with the codeword, second line starting with “//”.

Remove 2 for “(“ and another 2 for country Code \*/

MaxCCNameLength= 35-MXCodeLength + 33 – 2 – 2 – Length(MXTownName)

/\* IF Name is translated and Country is absent then the structure is “/CodeWord/Name” using max 2 lines of 35 char in MT \*/

MaxNameLength = 35 – MXCodeLength + 33

**IF** MXBICFI **IsPresent** THEN

MTAgent = MXBICFI

**ELSEIF** MXName **IsPresent** THEN

**IF** MXCountry **IsPresent** THEN

**IF** **Length**(MXName)> MaxCCNameLength THEN

MTAgent = **Concatenate**(Substring(MXName, 1, MaxCCNameLength - 1), “+”)

**ELSE**

MTAgent = MXName

**ENDIF**

MTAgent = **Concatenate**(MTAgent, “(“,MXCountry,”(“, MXTownName)

**ELSE** /\* MXCountry is not present \*/

**IF** **Length**(MXName)> MaxNameLength THEN

MTAgent = **Concatenate**(Substring(MXName, 1, MaxNameLength - 1), “+”)

**ELSE**

MTAgent = MXName

**ENDIF**

**ENDIF**

**ELSEIF** MXClearingID **IsPresent** THEN

MTAgent = **Concatenate**(MXClearingID.ClearingSystemIdentification.Code, MXClearingID.MemberIdentification)

**ENDIF**

### 4.2.7 MX\_To\_MTBICNameAgent

**Name**

MX\_To\_MTBICNameAgent(MXAgent, MaxLength; MTAgent)

**Business description**

The function extracts the BICFI or the Name or the ClearingSystemMemberIdentification, if Name and BIC are absent, from the MX structure. Name and ClearingSystemMemberIdentification are truncated after the length passed as parameter and added “+” for truncation indication. Preference is given to BIC.

The ISO Clearing System Identification is not converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

**Format**

**Input**

MXAgent : BranchAndFinancialInstitutionIdentification6

MaxLength : integer

**Output**

MTAgent : string

**Preconditions**

None.

**Formal description**

/\* Local variables :

MXBICFI, MXName : string

MXClearingID : ClearingSystemMemberIdentification2 \*/

MXBICFI = MXAgent.FinancialInstitutionIdentification.BICFI

MXName = MXAgent.FinancialInstitutionIdentification.Name

MXClearingID = MXAgent.FinancialInstitutionIdentification.ClearingSystemMemberIdentification

**IF** MXBICFI **IsPResent** THEN

MTAgent = MXBICFI

**ELSEIF** MXName **IsPresent** THEN

**IF Length**(MXName)> MaxLength THEN

MTAgent = **Concatenate**(**Substring**(MXName, 1, MaxLength - 1), “+”)

**ELSE**

MTAgent = MXName

**ENDIF**

**ELSEIF** MXClearingID **IsPresent** THEN

MTAgent = **Concatenate**(MXClearingID.ClearingSystemIdentification.Code, MXClearingID.MemberIdentification)

**IF Length**(MTAgent)> MaxLength THEN

MTAgent = Concatenate(Substring(MTAgent, 1, MaxLength - 1), “+”)

**ENDIF**

**ENDIF**

### 4.2.8 MX\_To\_MTFinancialInstitutionNameAndUnstructuredAddress

**Name**

MX\_To\_MTFinancialInstitutionNameAndUntructuredAddress

**Business description**

The function translates an MX FinancialInstitution Name and PostalAddress.AddressLine to MT NameAndAddress format (4\*35). In the MX identification typed *BranchAndFinancialInstitutionIdentification, if the BICFI is absent, the Name and Postal Address are mandatory for crossborder transactions. In order to cater for wider cases (eg camt messages), if Name is absent, dummy value “NOTPROVIDED” is used. The Postal Address must be structured if the payment is initated in MX or can be unstructured (i.e., AddressLine)* if the payment is initiated in MT. This function translates a Name and unstructured address (AddressLine). If the AddressLine uses 3 lines and the Name is longer than 35 char, it will be truncated after 34 characters (and sign “+” added). If the AddressLine uses only 2 lines or less then the 70 first characters of Name will be copied on line 1 and line 2 (if needed). Truncation is possible. A sign”+” is added at the end of the data if truncation is needed.

AddressLine is translated only if the value is different from “NOTPROVIDED” in AddressLine[1].

**Format**

**MX\_To\_MTFinancialInstitutionNameAndUnstructuredAddress**(MXAgent ; MTNameAndAddress)

**Input**

MXAgent: the entire structure of the MX agent typed *BranchAndFinancialInstitutionIdentification6*.

**Output**

MTNameAndAddress: name and address of the financial institution in the MT format (4\*35x).

**Preconditions**

None

**Formal description**

/\* Throughout the function, if translation of the source

component is spread over more than one line of the 4\*35x format, a Carriage Return Line Feed (*CRLF*) will be added

between consecutive lines to comply with the format of an MT field with multiple lines \*/

/\* Local variables

String: MXName,

MXNameAndAddress[]: table of string max 35 characters

Integer: NumberConsumedLines, AddressNumberOfLines \*/

AddressNumberOfLines = 0

MXName = MXAgent.FinancialInstitutionIdentification.Name

**IF Length**(MXName)< 1 THEN

MXName = “NOTPROVIDED”

**ENDIF**

**IF** MXAgent.FinancialInstitutionIdentification.PostalAddress.AddressLine **IsPresent** THEN

AddressNumberOfLines = NumberOfOccurrences(AddressLine)

**ENDIF**

/\* Limit Name depending on Address Line \*/

**IF** AddressNumberOfLines = 3 THEN

/\* Name max 35 char \*/

**IF Length**(MXName)> 35 THEN

MXName = **Concatenate**(**Substring**(MXName, 1, 34), “+”)

**ENDIF**

**ELSE**

/\* Name max 70 char \*/

**IF Length**(MXName)> 70 THEN

MXName = **Concatenate**(**Substring**(MXName, 1, 69), “+”)

**ENDIF**

**ENDIF**

**IF Length**(MXName)< 36 THEN

MXNameAndAddress[1] = **Substring**(MXName, 1, 35)

NumberConsumedLines = 1

**ELSE**

/\* MXName is max 70 char \*/

MXNameAndAddress[1] = **Substring**(MXName, 1, 35)

MXNameAndAddress[2] = **Substring**(MXName, 36)

NumberConsumedLines = 2

**ENDIF**

**IF** MXAgent.FinancialInstitutionIdentification.PostalAddress.AddressLine **IsPresent** AND MXAgent.FinancialInstitutionIdentification.PostalAddress.AddressLine[1] **Not Equal** to “NOTPROVIDED” THEN

**For i** = 1 to **NumberOfOccurrences** (MXAgent.FinancialInstitutionIdentification.PostalAddress.AddressLine)

**IF** **Length**(MXAgent.FinancialInstitutionIdentification.PostalAddress.AddressLine[i]) > 0 THEN

NumberConsumedLines = NumberConsumedLines +1

MXNameAndAddress[NumberConsumedLines]= MXAgent.PostalAddress.AddressLine[i]

**ENDIF**

**END loop**

**ENDIF**

/\* Copy to MT structure \*/

**For i=1** to NumberConsumedLines

**AppendToNextLine**(MXNameAndAddress[i], MTNameAndAddress)

**END LOOP**

### 4**.2.9 MX\_To\_MTFinancialInstitutionNameAndStructuredAddress**

**Name**

MX\_To\_MTFinancialInstitutionNameAndStructuredAddress

**Business description**

This function translates a MX Name and structured Postal Address for a Financial Institution to a MT Name and Address. The logic is similar to the one applied in MX\_To\_MTPartyNameAndStructuredAddress but the presence of LEI is not taken into account and the MT Name and Address is not structured with Numbers (as for 50F or 59F) for Financial Institutions.

The information will be translated following the order:

1. Name
2. StreetName, BuildingNumber, BuildingName, Floor, PostBox, Room, Department, SubDepartment
3. Country/TownName, PostCode, Country SubDivision, TownLocationName, DistrictName

If an element is missing there will be no indication it is missing in the MT string (ie., the position in the MT string will be insufficient to identify unambiguously the type of information)

The separator Comma will be used to separate the element in the MT string except between Country and TownName where “/” is used to be aligned with the MT structure for Line 3.

Country and TownName are mandatory in CBPR+ if structured postal address is used.

The Name will be truncated after 70 characters (“+” sign on position 70, if truncation is needed).

Case where 3 lines are available and StreetName line is present then Country Line can use up to 2 lines if needed otherwise StreetName line can use up to 2 lines if needed.

Case where 2 lines are available, StreetName line will use 1 line and Country line will use 1 line.

If StreetName line is absent, then Country line can use 2 lines.

The function handles the case where only Name is present.

**Format**

**MX\_To\_MTFinancialInstitutionNameAndStructuredAddress**(MXAgent; MTNameAndAddress )

**Input**

MXAgent: the entire structure of the MXAgent typed *BranchAndFinancialInstitutionIdentification6*.

**Output**

**MTNameAndAddress : max 4 lines of 35 char**

**Preconditions**

None

**Formal description**

/\*Throughout the function, if translation of the source

component is spread over more than one line of the 4\*35x format, a Carriage Return Line Feed (*CRLF*) will be added

between consecutive lines to comply with the format of an MT field with multiple lines\*/

/\* Local variables

i, NextIndex, NumberOfNameOccurrences : integer

Separ : string

MXNumber1, MXNumber2, MXNumber3 :string

MXTable2[], MXTable3[], MTNameAndAddressTable[] : table of string

NumberRemainingLines : Integer

MXNumber2Present : Boolean \*/

/\* To shorten the path, the following shortcut will be used \*/

MXPostalAddress = MXAgent.FinancialInstitutionIdentification.PostalAddress

MXTable2[1] = MXPostalAddress.StreetName

MXTable2[2] = MXPostalAddress.BuildingNumber

MXTable2[3] = MXPostalAddress.BuildingName

MXTable2[4] = MXPostalAddress.Floor

MXTable2[5] = MXPostalAddress.PostBox

MXTable2[6] = MXPostalAddress.Room

MXTable2[7] = MXPostalAddress.Department

MXTable2[8] = MXPostalAddress.SubDepartment

MXTable3[1] = MXPostalAddress.Country

MXTable3[2] = MXPostalAddress.TownName

MXTable3[3] = MXPostalAddress.PostCode

MXTable3[4] = MXPostalAddress.CountrySubdivision

MXTable3[5] = MXPostalAddress.TownNameLocation

MXTable3[6] = MXPostalAddress.DistrictName

Separ = “,”

MXNumber1 = MXAgent.financialInstitutionIdentification.Name

**IF** **Length**(MXNumber1) = 0 THEN

MXNumber1 = “NOTPROVIDED”

**ENDIF**

/\* 2 lines can be used for the name if needed \*/

**IF** **Length**(MXNumber1)> 70 THEN

MTNameAndAddressTable[1]= **Substring**(MXNumber1,1,35)

MTNameAndAddressTable[2]= **Concatenate**(**Substring**(MXNumber1,36,34), “+”)

NumberRemainingLines = 2

**ELSEIF Length**(MXNumber1)> 35 THEN

MTNameAndAddressTable[1]= **Substring**(MXNumber1,1,35))

MTNameAndAddressTable[2]= **Substring**(MXNumber1,36)

NumberRemainingLines = 2

**ELSE**

MTNameAndAddressTable[1]= MXNumber1

NumberRemainingLines = 3

**ENDIF** /\* Length(MXNumber1) \*/

**IF** MXPostalAddress.Country **NOT IsPresent** THEN

/\* No structured Postal address. Copy the Name to MTNameandAddress \*/

**For i** = 1 to (4 – NumberRemainingLines)

**IF Length**(MTNameAndAddressTable[i])> 0 THEN

**AppendToNextLine**(MTNameAndAddressTable[i], MTNameAndAddress)

**ELSE**

EXIT loop

**ENDIF**

**END loop**

Exit function

**ENDIF**

/\* Fill in MXNumber2 respecting the following order StreetName, BuildingNumber, BuildingName, Floor, PostBox, Room, Department, SubDepartment and using the separator between elements. Missing information is not indicated. \*/

**For i** = 1to 8

**{**

**IF** **Length**(MXTable2[i])> 0 THEN

**IF** **Length**(MXNumber2) = 0 THEN

/\* it is the first element found \*/

MXNumber2 = MXTable2[i]

**ELSE**

MXNumber2 = **Concatenate** (MXNumber2,Separ, MXTable2[i])

**ENDIF**

**ENDIF**

**}**

**Next i**

/\* Fill in MXNumber3 respecting the following order Country, TownName, PostCode, CountrySubDivision, TownLocationName, DistrictName and using the separator between elements except between Country and TownName where “/” is used. In CBPR+, if Structured postal address is used, Country and TownName are mandatory \*/

/\* Start with Country and TownName \*/ \*/

MXNumber3 = Concatenate(MXTable3[1], “/”)

MXNumber3 = Concatenate(MXNumber3, MXTable3[2])

/\* Continue with other elements \*/

**For i** = 3 to 6

**{**

**IF** **Length**(MXTable3[i])> 0 THEN

MXNumber3 = **Concatenate** (MXNumber3,Separ, MXTable3[i])

**ENDIF**

**}**

**Next i**

/\* Check if MXNumber2 has meaningful information \*/

**IF** **Length**(MXNumber2) > 0 THEN

MXNumber2Present = “true”

**ELSE**

MXNumber2Present = “false”

**ENDIF**

**Case 1**

**IF** NumberRemainingLines = 3 THEN

NextIndex = 4 – NumberRemainingLines + 1

**IF** MXNumber2Present THEN

/\* MXNumber3 can use 2 lines, if needed, else MXNumber2 can use 2 lines if needed \*/

**Call** Subfunction **Case11** /\* described below \*/

**ELSE**

/\* MXNumber2 is absent, 3 lines can be used by MXNumber3 \*/

**Call** Subfunction **Case12** /\* described below \*/

**ENDIF**

**ENDIF**

**Case 2**

**IF** NumberRemainingLines = 2 THEN

NextIndex = 4 – NumberRemainingLines + 1

**IF** MXNumber2Present THEN

/\* MXNumber2 uses 1 line and MXNumber3 uses 1 line \*/

**Call** Subfunction **Case21**

**ELSE**

/\* MXNumber3 can use 2 lines \*/

**Call** Subfunction **Case22**

**ENDIF**

**ENDIF**

/\* Fill in the MT structure with CRLF between the lines \*/

**For i** = 1 to 4

**IF Length(**MTNameAndAddressTable[i])> 0 THEN

**AppendToNextLine**(MTNameAndAddressTable[i], MTNameAndAddress)

**ELSE**

EXIT loop

**ENDIF**

**Next i**

**Subfunction definitions**

**Subfunction Case11**

/\* MXNumber3 can use 2 lines, if needed, else MXNumber2 can use 2 lines if needed \*/

**IF** **Length**(MXNumber3)> 70 THEN

/\* MXNumber3 uses 2 lines, MXNumber2 uses 1 line \*/

/\* Fill in with MXNumber2 \*/

**IF** **Length**(MXNumber2) > 35 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber2,1,34), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]=MXNumber2

**ENDIF**

NextIndex = NextIndex + 1

/\* Fill in 2 lines with MXNumber3\*/

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35)

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber3,36,34), “+”)

**ELSEIF Length**(MXNumber3)> 35 THEN

/\* MXNumber 3 uses 2 lines but no truncation needed for MXNumber3. MXNumber2 uses 1 line \*/

/\* Fill in with MXNumber2 \*/

**IF** **Length**(MXNumber2) > 35 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber2,1,34), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]=MXNumber2

**ENDIF**

NextIndex = NextIndex + 1

/\* Fill in 2 lines with MXNumber3\*/

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,36)

**ELSE**

/\* LENGTH(MXNumber3)<= 35 uses 1 line , MXNumber2 can use 2 lines if needed \*/

/\* Fill in with MXNumber2 \*/

**IF** **Length**(MXNumber2) > 70 THEN

MTNameAndAddressTable[NextIndex]=Substring(MXNumber2, 1, 35)

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber2, 36, 34), “+”)

NextIndex = NextIndex + 1

**ELSEIF** **Length**(MXNumber2) > 35 THEN

MTNameAndAddressTable[NextIndex]=Substring(MXNumber2, 1, 35)

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber2, 36)

NextIndex = NextIndex + 1

**ELSE** /\* MXNumber2 uses 1 line \*/

MTNameAndAddressTable[NextIndex]= MXNumber2

NextIndex = NextIndex + 1

**ENDIF**

MTNameAndAddressTable[NextIndex]= MXNumber3

**ENDIF** /\* ENDIF Length(MXNumber3)> 70 \*/

**Subfunction** **Case12**

/\* MXNumber2 is absent, 3 lines can be used by MXNumber3 \*/

**IF** **Length**(MXNumber3)> 105 THEN

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35)

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,36,35))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber3,71,34), “+”)

**ELSEIF** **Length**(MXNumber3)> 70 THEN

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]=**Substring**(MXNumber3,36,35))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,71)

**ELSEIF** **Length**(MXNumber3)> 35 THEN

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35))

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,36)

**ELSE**

MTNameAndAddressTable[NextIndex]= MXNumber3

**ENDIF**

**Subfunction** **Case21**

/\* MXNumber2 uses 1 line and MXNumber3 uses 1 line \*/

**IF** **Length**(MXNumber2) > 35 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber2,1,34), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]= MXNumber2

**ENDIF**

NextIndex = NextIndex + 1

**IF** **Length**(MXNumber3) > 35 THEN

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber3,1,34), “+”)

**ELSE**

MTNameAndAddressTable[NextIndex]= MXNumber3

**ENDIF**

**Subfunction** **Case22**

/\* MXNumber3 can use 2 lines \*/

**IF** **Length**(MXNumber3) > 70 THEN

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35)

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Concatenate**(**Substring**(MXNumber3,36,34), “+”)

**ELSEIF** LENGTH(MXNumber3) > 35 THEN

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,1,35)

NextIndex = NextIndex + 1

MTNameAndAddressTable[NextIndex]= **Substring**(MXNumber3,36)

**ELSE**

MTNameAndAddressTable[NextIndex]= MXNumber3

**ENDIF**

### 4.2.10 MX\_To\_MT54A

**Name**

MX\_To\_MT54A

**Business description**

The function extracts from InstructionForNextAgent the code /FIN54/ coming from very specific scenarios in MT. Due to the definition of pacs.009 ADV, this function is not relevant anymore and will be deleted.

It is expected that in the next leg of the transaction, the field has been removed as it is aimed for the receiver of the MT message. Nevertheless the translation is provided.

MX\_To\_MT54A (MXInstructionForNextAgent; MT54A)

**Input**

MXInstructionForNextAgent typed InstructionForNextAgent1

**Output**

MT54A: Field54 format A, Identifier Code (BIC)

**Preconditions**

None

**Formal description**

/\* Local variables

MTInstruction, MT54BIC : string

GenericPattern : pattern (regular expression) \*/

/\* To extract /FIN54/ and related business information which is expected to be a valid BIC, search for the codeword /FIN54/ and extract them. It is assumed that the codeword /FIN54/ is present only once. Other code words present in the string must have a pattern “/max8c/” eg., /REC/, /INT/, /INTA/.\*/

GenericPattern = /[A-Z0-9]{1,8}/

/\* The pattern has to take into account the FIN code words but also the 2 new code words /FIN54/ and /FIN53/ making digits also possible in the pattern \*/

/\* First build a string by concatenation of the 4 possible occurrences from MXInstructionForNextAgent and scan the string \*/

MTInstruction = “”

**For n**=1 to 4

/\*Build one string \*/

**IF** InstructionForNextAgent[n].InstructionInformation **IsPresent** THEN

MTInstruction = Concatenation(MTInstruction, InstructionForNextAgent[n].InstructionInformation)

**ENDIF**

**End loop**

/\*Search for pattern **/FIN54/** in MTInstruction.

If found, extract the pattern and the following related information up to the next pattern, if any or up to the end in case no next pattern is found. Store the pattern and related information in MT54BIC\*/

**IF NOT IsPresentPattern**(MTInstruction, “/FIN54/”) THEN

EXIT function

**ENDIF**

MT54BIC = **ExtractBetweenPattern**(MTInstruction, ”/FIN54/”, GenericPattern)

**IF Length**(MT54BIC) < 1 THEN

EXIT function

**ENDIF**

**IF** MT54BIC **IsValidBIC** THEN

/\*Valid BIC means that the BIC exists. This implies to look up in a table maintaining BIC data \*/

MT54A = MT54BIC

**ELSE**

/\* Ignore and do not create Field 54A in MT\*/

T20028 /\* Error code described in the error code list \*/

**ENDIF**

### 4.2.11 MX\_To\_MT53A

/\*Internal note. This function is the same as MX\_To\_MT54A but the business description and usage is different. \*/

**Name**

MX\_To\_MT53A

**Business description**

The function extracts from InstructionForNextAgent the code /FIN53/ coming from a very specific scenario in MT.

More information about the origin of the /FIN53/ can be found in METAFCT001 described in the excel document MT103 to pacs.008 translation (and also in METAFCT002 MT202 CORE to pacs.009; METAFCT003 MT202 COVE to pacs.009).

It is expected that in the next leg of the transaction, the field has been removed as it is aimed for the receiver of the MT message. Nevertheless the translation is provided.

MX\_To\_MT53A must be maintained together with SubfunctionInstructionForNextAgent. The BIC is extracted by MX\_To\_MT53A only if the codeword /FIN53/ is followed by an exact BIC. If the structure is “/FIN53/BICText” or “/FIN53/BIC Text”, then the information is considered as being an invalid BIC and therefore not translated to 53A. In order not to loose information, the codeword and following information is therefore translated to Field 72 by the SubfunctionInstructionForNextAgent

Note that it is not expected to have code /FIN53/ present several times. In the assumption this happens only the first occurrence is checked to extract a valid BIC.

MX\_To\_MT53A (MXInstructionForNextAgent; MT53A)

**Input**

MXInstructionForNextAgent typed InstructionForNextAgent1

**Output**

MT53A: Field53 format A, Identifier Code (BIC)

**Preconditions**

None

**Formal description**

/\* Local variables

MTInstruction, MT53BIC : string

GenericPattern : pattern (regular expression)

TempREC : string

m : integer \*/

TempREC = “/TempREC/”

GenericPattern = /[A-Z0-9]{1,8}/

/\* The pattern has to take into account the FIN code words but also the new code word /FIN53/ making digits also possible in the pattern \*/

/\* To extract /FIN53/ and related business information which is expected to be a valid BIC, search for the codeword /FIN53/ and extract them. It is assumed that the codeword /FIN53/ is present only once. Other code words present in the string must have a pattern “/max8c/” eg., /REC/, /INT/, /INTA/. \*/

/\* First build a string by concatenation of the 6 possible occurrences in MXInstructionForNextAgent in the same way it is built in SubfunctionInstructionForNextAgent and scan the string \*/

MTInstruction = “”

**IF** MXInstructionForNextAgent[1] **IsPresent** THEN

MTInstruction = MXInstructionForNextAgent[1].InstructionInformation

**For m** = 2 to 6

**IF** MXInstructionForNextAgent[m] **IsPresent** THEN

**IF**(**Length**(MXInstructionForNextAgent[m-1].InstructionInformation)< 35 AND MXInstructionForNextAgent[m].InstructionInformation **DOES NOT** **START with** “/GenericMax8c/”) THEN

MTInstruction = **Concatenate** (MTInstruction, **TempREC**, MXInstructionForNextAgent[m].InstructionInformation)

**ELSE**

MTInstruction = **Concatenate** (MTInstruction, MXInstructionForNextAgent[m].InstructionInformation)

**ENDIF**

**ENDIF**

**Next** m

**ENDIF**

/\*Search for pattern **/FIN53/** in MTInstruction.

If found, extract the pattern and the following related information up to the next pattern, if any or up to the end in case no next pattern is found. Store the pattern and related information in MT53BIC\*/

**IF NOT IsPresentPattern**(MTInstruction, “/FIN53/”) THEN

EXIT function

**ENDIF**

MT53BIC = **ExtractBetweenPattern**(MTInstruction, ”/FIN53/”, GenericPattern)

**IF Length**(MT53BIC) < 1 THEN

EXIT function

**ENDIF**

**IF** MT53BIC **IsValidBIC** THEN

/\*Valid BIC means that the BIC exists. This implies to look up in a table maintaining BIC data \*/

MT53A = MT53BIC

**ELSE**

/\* Ignore and do not create Field 53A in MT\*/

**ENDIF**

### 4.2.12 MX\_To\_MTAgentGeneric

The function is used to translate all agents and should cover all cases for payments initiated in MX but also payments initiated in MT. In MX, an agent must be identified with at least a BICFI or Name and PostalAddress for cross border payments. For domestic payments, a ClearingSystemMemberID alone is allowed. Depending on MX elements present and target field in MT, the Agent MT option (ie A, B, C or D) is defined. If a BICFI is present, it is always translated with the highest priority (ie option A).

**Name**

MX\_To\_MTAgentGeneric

**Business description**

**Input**

PathAgent\_In typed BranchAndFinancialInstitutionIdentification6

PathAccount\_In typed CashAccount38 (optional)

PathAgent\_Out : [SeqLetter/]MT Tag structure

IsAllowedOptionB : boolean

IsAllowedOptionC : boolean

IsAllowedClearingChannel : boolean

PathClearingChannel typed ClearingChannel2Code (optional)

MT : MT message

**Output**

MT (ie Elements already entered in the MT message)

Note : IsAllowedClearingChannel is “true” for IntermediaryAgent1 and CreditorAgent. Must be “false” for all the other agents.

If IsAllowedClearingChannel is true, that means that if the ClearingChanel = “RTGS” then it can be translated like “//RT” or “//FW” in option A (BIC present) or “//RT” can be concatenated with the MT ClearingSystemMemberID in option B,C or D.

**Preconditions**

PathAgent\_In is not empty

**Formal description**

/\* Local variable \*/

/\* IMPORTANT : the order of the IF ELSEIF..statement is important as it defines priority when the cases are not exclusive. For example in option B,C and D, translation priority is ClearingSystemMemberID then account then ClearingChannel alone \*/

**IF IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.BICFI) THEN

**Call SubfunctionBIC**

/\* Subfunction defined below \*/

**ELSEIF**

**IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.ClearingSystemMemberIdentification) AND

**Length**(PathAgent\_In.FinancialInstitutionIdentification.Name)<= 35 AND

**Length**(PathAgent\_In.FinancialInstitutionIdentification.Name)> 0 AND

(PathAgent\_In.FinancialInstitutionIdentification.Name =

PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1])AND

PathAgent\_In.FinancialInstitutionIdentification.Name <> “NOTPROVIDED”

AND Substring(PathAgent\_In.FinancialInstitutionIdentification.Name,1,2) =”//”

AND IsAbsent(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[3])

/\* This case can only happen if the payment is originated in MT with field 5xa with option B or C containing ClearingSystemCode with no equivalence in ISO list. A workaround was introduced to still translate the ClearingSystemMemberId (ie National Sort code) as it is the only identifier for the agent (in option A and D, it is not translated as the BIC or Name and Address can be used to identify the agent). The workaround in MT to MX consists in copying the ClearingSystem including “//” both in Name and first line of AddressLine. The MT Location if present is then translated to AddressLine[2]of AddressLine The translation below restores the translation back to option B or C. The presence of the ClearingChannel with value "RTGS" is ignored (see SubfunctionBIC to llustrate how the ClearingChannel influences the translation of ClearingSystemMemberID) in order to simplify the translation in this workaround case. If later there is a need to add it, the subfunction below will be amended \*/

**Call SubfunctionNoEquivalentMXClearing**

**ELSEIF**

**IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.ClearingSystemMemberIdentification) AND

**{IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.Name) OR

**(**PathAgent\_In.FinancialInstitutionIdentification.Name = “NOTPROVIDED” AND

**IsPresent(**PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1] AND

**IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[2] **)**

**}**

/\* IF Name is absent then PostalAddress must be absent according to CBPR+ UG rules. IF Name = “NOTPROVIDED”, it may come from translation option B or C to MX and therefore AddressLine must also be equal to “NOTPROVIDED” unless a MT location was present. Only the first occurrence of AddressLine can be present for this scenario. If Account is present, it is not translated. \*/

/\* Note that in camt messages, there is no rules in the Agent. So Name can be absent with AdressLine[1,2,3] present. In that case the above criteria are not fully correct and translation to option D should take place but is not foreseen here. Ths makes the translation rules very complex and **we do the assumption that in camt messages, Agents will be used in the same way as in pacs messages ie if AddressLine is present then Name is present too**. \*/

**Call SubfunctionOnlyClearingSystemMemberID**

**ELSEIF**

**IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.ClearingSystemMemberIdentification)

AND

**IsPresent**(PathAccount\_In**}**

AND

**{**

IF PathAgent\_In.FinancialInstitutionIdentification.Name = “NOTPROVIDED”)AND

**IsPresent(**PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1] AND

**IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[2]

**}**

/\* Name with value “NOTPROVIDED” is a side effect of MT to MX translation Option B and C with PartyIdentifier being an account. Otherwise if the payment is created in MX, Agent is mandatory if account is present. In this scenario, Name is mandatory. Dummy value “NOTPROVIDED” is only expected from MT to MX translation. \*/

**Call SubfunctionOnlyAccount**

**ELSEIF**

**IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.ClearingSystemMemberIdentification) AND

**IsAbsent**(PathAccount\_In) AND

PathAgent\_In.FinancialInstitutionIdentification.Name = “NOTPROVIDED”)AND

AND **IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[2]) AND

**{(IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]) AND PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]<> “NOTPROVIDED” **)** **OR** (IsAllowedClearingChannel AND PathClearingChannel = “RTGS” AND PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]= “NOTPROVIDED” )**}** THEN

/\* Scenario resulting from translation MT to MX (MT\_To\_MXClearingSystemToNameAndAddressLine) with option B with only location and no PartyIdentifier is presentOR partyIdentifier being only the clearing channel //RT with or without a location. This should come from cases:

Option B with location, Option B with “//RT” and location, Option B with “//RT”, Option C with “//RT”. \*/

**Call** **SubfunctionExceptionCase**

**ELSE**

**Call SubfunctionNameAndAddress**

**ENDIF**

**/\* Subfunctions definition \*/**

**SubfunctionBIC**

PathAgent\_In Translate To [**MX\_To\_MTBICFI**]PathAgent\_Out.OptionA.IdentifierCode

**IF IsPresent**(PathAccount\_In)THEN

PathAccount\_In Translate To [**MX\_To\_MTAccount**]PathAgent\_Out.OptionA.PartyIdentifier

**ELSEIF**

IsAllowedClearingChannel AND PathClearingChannel = “RTGS” THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierAndChannel**]PathAgent\_Out.OptionA.PartyIdentifier

**ELSEIF**

**IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.ClearingSystemMemberIdentification)THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifier**]PathAgent\_Out.OptionA.PartyIdentifier

**ENDIF**

**/\* End SubfunctionBIC \*/**

**SubfunctionNoEquivalentMXClearing**

/\*Translate to option B or C if it is allowed by the type of agent \*/

**IF IsAbsent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[2])THEN

**IF** IsAllowedOptionC THEN

PathAgent\_In Translate To [**MX\_To\_MTNameAndAddressToClearingSystemIdentifier**]PathAgent\_Out.OptionC.PartyIdentifier

**ELSEIF** IsAllowedOptionB THEN

PathAgent\_In Translate To [**MX\_To\_MTNameAndAddressToClearingSystemIdentifier**]PathAgent\_Out.OptionB.PartyIdentifier

**ELSE** T20197

/\* Due to the fact that this case can only occur in translation from option B and C, translation back to MT of this configuration expects one of these options is allowed \*/

**ENDIF**

**ELSEIF**

/\* AddressLine[2] is present \*/

**IF** IsAllowedOptionB THEN

PathAgent\_In Translate To [**MX\_To\_MTNameAndAddressToClearingSystemIdentifier**]PathAgent\_Out.OptionB.PartyIdentifier

PathAgent\_Out.OptionB.Location = PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[2]

/\* Nothing is expected after AdressLine[2] per MT to MX translation \*/

**ELSE** T20197

**ENDIF**

**ELSE** T20197

**ENDIF**

**/\* End SubfunctionNoEquivalentMXClearing \*/**

**SubfunctionOnlyClearingSystemMemberID**

/\* The subfunction translates the ClearingSystemMemberID to option C or B if it is allowed by the type of agent. If not allowed (ie., DebtorAgent) then the ClearingSystemMemberID is translated to option D and copied also in Subfield 2 Name and Address with “//” included to identify it is a clearing system member ID. The subfunction also takes into account the Clearing channel = “RTGS” and if the ClearingChannel is allowed with the type of agent \*/

/\* Instantiate **MX\_To\_MTClearingIdentifierGeneric** with the function to be used \*/

**IF** IsAllowedClearingChannel AND PathClearingChannel = “RTGS” THEN

/\* case 56a and 57a \*/

**MX\_To\_MTClearingIdentifierGeneric** = **MX\_To\_MTClearingIdentifierAndChannel**

**ELSE**

**MX\_To\_MTClearingIdentifierGeneric** = **MX\_To\_MTClearingIdentifier**

/\* Clearing channel not allowed in the MT ClearingSystemMemberID ID or value different from “RTGS”. ClearingChannel is not allowed for 52a, 53a, 54a, 55a. \*/

**ENDIF**

**IF IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1])AND PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]<> “NOTPROVIDED” THEN

/\* Only from MT to MX if option B with Location \*/

**IF** IsAllowedOptionB **THEN**

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierGeneric**]PathAgent\_Out.OptionB.PartyIdentifier

Copy (PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]) To PathAgent\_Out.OptionB.Location

**ELSE** T20197

/\* Option B not allowed \*/

**ENDIF**

**ELSE** /\* no AddresseLine [1] \*/

**IF** IsAllowedOptionC THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierGeneric**]PathAgent\_Out.OptionC.PartyIdentifier

**ELSEIF** IsAllowedOptionB THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierGeneric**]PathAgent\_Out.OptionB.PartyIdentifier

**ELSE**

/\*– neither option B nor C is allowed. For example 52a \*/

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierGeneric**]PathAgent\_Out.OptionD.PartyIdentifier

/\* Copy the MT ClearingSystemMemberID (ie National Sort Code) as such with “//” in subfield 2 Name And Address to get a valid MT \*/

PathAgent\_Out.OptionD.NameAndAddress[1] = PathAgent\_Out.OptionD.PartyIdentifier

**ENDIF**

**ENDIF** /\* End AddressLine[1] \*/

**/\* End SubfunctionOnlyClearingSystemMemberID \*/**

**SubfunctionOnlyAccount**

**IF IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1])AND PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]<> “NOTPROVIDED”

**IF** IsAllowedOptionB **THEN**

PathAccount\_In Translate To [**MX\_To\_MTAccount**]PathAgent\_Out.OptionB.PartyIdentifier

PathAgent\_Out.OptionB.Location = PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]

**ELSE** T20197

**ENDIF**

**ELSE**

**IF** IsAllowedOptionC **THEN**

PathAccount\_In Translate To [**MX\_To\_MTAccount**]PathAgent\_Out.OptionC.PartyIdentifier

**ELSEIF** IsAllowedOptionB **THEN**

PathAccount\_In Translate To [**MX\_To\_MTAccount**]PathAgent\_Out.OptionB.PartyIdentifier

**ELSE** T20197

**ENDIF**

**ENDIF**

**SubfunctionNameAndAddress**

/\* Order in the IF THEN ELSE statement is important as it defines the translation priorities \*/

**IF IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine) THEN

PathAgent\_In Translate To [**MX\_To\_MTFinancialInstitutionNameAndUnstructuredAddress**] PathAgent\_Out.OptionD.NameAndAddress

**ELSEIF**

**IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.CountryCode) THEN

PathAgent\_In Translate To [**MX\_To\_MTFinancialInstitutionNameAndStructuredAddress**] PathAgent\_Out.OptionD.NameAndAddress

**ELSEIF**

/\* It is important that the cases where Name is “NOTPROVIDED” are treated above in that order as they are potential cases eligible for translation to option B or C, “NOTPROVIDED” being a dummy value used in MT to MX to meet cross border rule where either BIC or Name and Address must be present – Domestic payments being an exception in the CBPR+ payments, the same rule will apply ie., to use NOTPROVIDED as default for Name And Address \*/

**IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.Name) THEN

PathAgent\_In Translate To [**MX\_To\_MTFinancialInstitutionNameAndUnstructuredAddress**] PathAgent\_Out.OptionD.NameAndAddress

/\* Name alone should not occur in CBPR+ UG. Case covered for smooth translation \*/

**ELSE** Copy "NOTPROVIDED" to PathAgent\_Out.OptionD.NameAndAddress

/\* Case covered for smooth translation. It should not occur if CBPR+ UG rules are applied \*/

**ENDIF**

/\* Fill in the PartyIdentifier subfield \*/

**IF IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.ClearingSystemMemberIdentification) THEN

**IF** IsAllowedClearingChannel AND PathClearingChannel = “RTGS” THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierAndChannel**] PathAgent\_Out.OptionD.PartyIdentifier

**ELSE**

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifier**] PathAgent\_Out.OptionD.PartyIdentifier

**ENDIF**

**ELSEIF**

**IsPresent**(PathAccount\_In)THEN

PathAccount\_In Translate To [**MX\_To\_MTAccount**]PathAgent\_Out.OptionD.PartyIdentifier

**ELSEIF**

/\* still translate the clearing channel= “RTGS” when allowed and when ClearingSystemMemberID is absent \*/

IsAllowedClearingChannel AND PathClearingChannel = “RTGS” THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierAndChannel**]PathAgent\_Out.OptionD.PartyIdentifier

/\* PartyIdentifier = “//RT” or “//FW” \*/

**ENDIF**

**/\* End SubfunctionNameAndAddress \*/**

**SubfunctionExceptionCase**

**IF IsPresent**(PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]AND (PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1]<> “NOTPROVIDED” THEN

**IF** IsAllowedOptionB THEN

Copy PathAgent\_In.FinancialInstitutionIdentification.PostalAddress.AddressLine[1] To

PathAgent\_Out.OptionB.Location

**IF** IsAllowedClearingChannel AND PathClearingChannel = “RTGS” THEN

/\* ClearingSystemMemberID is absent per priority in the IF statement \*/

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierAndChannel**] PathAgent\_Out.OptionB.PartyIdentifer

/\* PartyIdentifer = “//RT” \*/

**ENDIF**

**ELSE** T20197 /\* Option B not allowed \*/

**ENDIF**

**ELSE** /\* AddressLine[1] is absent is not possible as Name = “NOTPROVIDED”, then value = “NOTPROVIDED” And ClearingChannel = “RTGS” and IsAllowedClearingChannel is true \*/

**IF** IsAllowedOptionC THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierAndChannel**] PathAgent\_Out.OptionC.PartyIdentifer

/\* PartyIdentifer = “//RT” \*/

**ELSEIF** IsAllowedOptionB THEN

PathAgent\_In Translate To [**MX\_To\_MTClearingIdentifierAndChannel**] PathAgent\_Out.OptionB.PartyIdentifer

/\* PartyIdentifer = “//RT” \*/

**ELSE** T20197

**ENDIF**

## 4.3 Other Translation Rule Descriptions

The translation rule descriptions provided in this section are for translation rules that relate to any other fields or elements that are not specifically related to agents or customer parties.

### 4.3.1 MX\_To\_MTAccount

**Name**

MX\_To\_MTAccount

**Business description**

The function translates an MX account to an MT account. The MX account typed *CashAccount38 or CashAccount40* contains an account identification element and optional account type, currency and name elements. This function translates the MX account identification. Type, Currency, Name and Proxy elements are not translated.

**Format**

**MX\_To\_MTAccount**(MXAccount ; MTAccount)

**Input**

MXAccount: the entire structure of the MX account identification typed CashAccount38.

**Output**

MTAccount: the account in the MT format (/34x).

**Preconditions**

None.

**Formal description**

**IF** **IsPresent**(Identification.IBAN)THEN

MTAccount = **Concatenate**(“/”, Identification.IBAN)

**ELSEIF** **IsPresent**(Identification.Other.Identification)THEN

/\*Account is a local variable\*/

Account = Identification.Other.Identification

/\*Check whether Account is a CHIPS Universal Identifier (format CH123456) to be preceded by a double slash in an MT account field\*/

**IF**(Identification.Other.SchemeName.Code) = “CUID” AND

**Length**(Account) = 6

MTAccount = **Concatenate**(“//CH”, Account)

**ELSE**

MTAccount = **Concatenate**(“/”, Account)

**ENDIF**

**ENDIF**

**Example 1 MX Source:**

An MX DebtorAccount

<DbtrAcct>

<Id>

<Othr>

<Id>123456789</Id>

</Othr>

</Id>

</DbtrAcct>

**MT Translation:**

Subfield 1 of field :50a:/123456789

**Example 2 MX Source:**

An MX CreditorAccount(IBAN)

<CdtrAcct>

<Id>

<IBAN>BE16001216377774</IBAN>

</Id>

</CdtrAcct>

**MT Translation:**

Subfield 1 of field :59a:/BE16001216377774

**Example 3 MX Source:**

An MX CreditorAccount (CHIPS Universal Identifier)

<CdtrAcct>

<Id>

<Othr>

<Id>123456</Id>

<SchmeNm>

<Cd>CUID</Cd>

</SchmeNm>

</Othr>

</Id>

</CdtrAcct>

**MT Translation:**

Subfield 1 of field :59a://CH123456

### 4.3.2 MX\_To\_MTSettlementAccount

**Name**

MX\_To\_MTSettlementAccount

**Business description**

The function translates an MX SettlementMethod and MX SettlementAccount to an MT PartyIdentifier. The MX account typed *CashAccount38* or *CashAccount40*contains a mandatory account identification element and optional account type, currency and name elements. This function translates the MX account identification. Type, Currency and Name elements are truncated.

**Format**

**MX\_To\_MTSettlementAccount**(MXSettlementMethod, MXSettlementAccount ; MTPartyIdentifier)

**Input**

MXSettlementMethod: code indicating the MX settlement method.

MXSettlementAccount: the entire structure of the MX settlement account typed *CashAccount38*, *CashAccount40*

**Output**

MTPartyIdentifier: PartyIdentifier in the MT format [1!a]/34x where

[1!a] is an optional debit/credit indicator (D or C)

/34x is the account in MT format

**Preconditions**

None.

**Formal description**

/\*Translation of the account by calling a sub-function. Account is a local variable\*/

Account = **MX\_To\_MTAccount**(MXSettlementAccount)

/\*Check the code in MXSettlementMethod. MXSettlementAccount may

only be used with codes INGA or INDA\*/

**IF** MXSettlementMethod = “INGA” THEN

MTPartyIdentifier = **Concatenate**(“/C”, Account)

/\* Default value for INDA, so “/D” is not needed \*/

**ELSEIF** MXSettlementMethod = “INDA” THEN

MTPartyIdentifier = Account

**ENDIF**

**Example MX Source:**

MXSettlementMethod “INDA”

MX SettlementAccount

<SttlmAcct>

<Id>

<Othr>

<Id>123456789</Id>

</Othr>

</Id>

</SttlmAcct>

**MT Translation (to field 53B Sender’s Correspondent):**

:53B:/123456789

### 4.3.3 MX\_To\_MTDate

**Name**

MX\_To\_MTDate

**Business description**

The function translates an MX date expressed as “YYYY-MM-DD” to an MT date expressed as “YYYYMMDD” by deleting the separator “-“ between year, month and date.

**Format**

**MX\_To\_MTDate**(MXDate ; MTDate)

**Input**

MXDate: date in the MX format typed *ISODate* expressed as “YYYY-MM-DD”

**Output**

MTDate: date in the MT format expressed as “YYYYMMDD”

**Preconditions**

None.

**Formal description**

MTDate = **ReplacePattern**(MXDate, “-“, “”) **Example**

MX Source: 2007-01-01 MT Translation: 20070101

MT Source: 1987-01-01

MX Translation: 19870101

### 4.3.4 MX\_To\_MTTimeOffset

**Name**

MX\_To\_MTTimeOffset

**Business description**

The function translates an MX time and offset expressed as “HH:MM:SS+/-HH:MM” to an MT time and offset expressed as “HHMM+/-HHMM” by deleting the seconds SS and removing the “:“pattern in both source time and source offset. The seconds are truncated and not rounded up.

**Format**

**MX\_To\_MTTimeOffset**(MXTimeOffset ; MTTimeOffset)

**Input**

MXTimeOffset: time and offset in the MX format typed ISOTime, expressed as

“HH:MM:SS+/-HH:MM” where

HH:MM:SS is the time

+/- is the offset sign

HH:MM is the offset

**Output**

MTTimeOffset: time and offset in the MT format expressed as “HHMM+/-HHMM” where HHMM is the time

+/- is the offset sign

HHMM is the offset

**Preconditions**

None.

**Formal description**

/\*MTTime, MTOffsetSign and MTOffset are local variables\*/

MTTime = **ReplacePattern**(Substring(MXTimeOffset, 1, 5), “:”, “”)

MTOffset = **ReplacePattern**(Substring(MXTimeOffset, 10, 6), “:”, “”)

MTOffsetSign = **Substring**(MXTimeOffset, 9, 1)

/\*Build the output string\*/

MTTimeOffset = **Concatenate**(MTTime, MTOffsetSign , MTOffset) **Example**

MX Source: 13:55:00+01:00

MT Translation: 1355+0100

### 4.3.5 MX\_To\_MTDateTimeOffset

**Name**

MX\_To\_MTDateTimeOffset

**Business description**

The function translates an MX date and time with offset expressed as “YYYY-MMDDTHH:MM:SS+/-HH:MM” to an MT date expressed as YYYYMMDD and an MT time with offset expressed as “HHMM+/-HHMM” by:

* removing the “-“pattern between year, month and date in the MX date
* deleting the “T” separator between the MX date and time with offset
* deleting the seconds in the MX time and removing the “:“pattern between hours and minutes in the MX time and offset (the seconds are truncated and not rounded up).

**Format**

**MX\_To\_MTDateTimeOffset**(MXDateTime ; MTDate, MTTimeOffset)

**Input**

MXDateTime: date and time in the MX format typed *ISODateTime* expressed as

“YYYY-MM-DDTHH:MM:SS+/-HH:MM where YYYY-MM-DD is the date

T is the date/time separator

HH:MM:SS is the time

+/- is the offset sign

HH:MM is the time offset

**Output**

MTDate: date in the MT format expressed as “YYYYMMDD”.

MTTimeOffset: time and offset in the MT message expressed as “HHMM+/-HHMM” where HHMM is the time

+/- is the offset sign

HHMM is the time offset

**Preconditions**

None.

**Formal description**

/\*Sub-functions are called to translate the MX date and time with offset. MXDate and MXTimeOffset are local variables\*/ MXDate = **Substring**(MXDateTime, 1, 10)

MXTimeOffset = **Substring**(MXDateTime, 12)

/\*Translate date and time\*/

MTDate = **MX\_To\_MTDate**(MXDate)

MTTimeOffset = **MX\_To\_MTTimeOffset**(MXTimeOffset) **Example**

MX Source: 2007-02-22T13:55:00+01:00 MT Translation:

Date 20070222

Time with offset 1355+0100

### 4.3.6 MX\_To\_MTRate

**Name**

MX\_To\_MTRate

**Business description**

The function translates an MX (exchange) rate -including possible sign- to an MT (exchange) rate -without sign- by deleting the sign if present and replacing the decimal separator “.” by a “,”. If the MX rate is an integer, the decimal separator “,” is added at the end. Insignificant zeroes -if any- are also deleted.

**Format**

**MX\_To\_MTRate**(MXRate ; MTRate)

**Input**

MXRate: (exchange) rate in the MX format typed *BaseOneRate* with maximum 11 digits, maximum 10 fraction digits. The decimal separator “.” is optional and the rate can start with a “-“ or “+“ sign.

**Output**

MTRate: (exchange) rate in the MT format expressed as 12d (up to 12 characters including the mandatory decimal separator “,” without sign)

**Preconditions**

None.

**Formal description**

/\*The sign -if present- is removed. Rate is a local variable\*/

**IF Substring**(MXRate, 1, 1) = “-“ OR “+” THEN

Rate = **Substring**(MXRate, 2)

**ELSE**

Rate = MXRate

**ENDIF**

/\*Replace the decimal separator “.” -if present- by a “,”. If no decimal separator is present, addition of “,” at the end of the rate\*/

Rate = **ReplacePattern**(MXRate, “.”, “,”)

**IF** **IsPresentPattern**(Rate, “,”)THEN

/\*Delete insignificant zeroes at the left and right\*/

Rate = **TrimLeft**(Rate, “0”)

Rate = **TrimRight**(Rate, “0”)

/\*Check whether the rate has a strictly fractional part that is starts with a “,”. If this is the case, a zero is added before the comma as required in MT fields of the decimal format\*/

**IF** **Substring**(Rate, 1, 1) = “,” THEN

MTRate = **Concatenate**(“0”, Rate)

**ELSE**

MTRate = Rate

**ENDIF**

**ELSE**

/\*Remove insignificant zeroes -if any- at the left and add a decimal separator at the end to obtain the rate\*/

Rate = **TrimLeft**(Rate, “0”)

MTRate = **Concatenate**(Rate, “,”)

**ENDIF**

**Example**

MX Source: .1245

MT Translation: 0,1245

MX Source: +0.9876 MT Translation: 0,9876

MX Source: 010.3333 MT Translation: 10,3333

MX Source: -.124500 MT Translation: 0,1245

MX Source: 1

MT Translation: 1,

### 4.3.7 MX\_To\_MTCurrencyAmount

**Name**

MX\_To\_MTCurrencyAmount

**Business description**

The function translates an MX amount with currency embedded as an XML attribute to:

* an MT currency by extracting the currency of the XML attribute
* an MT amount by replacing the decimal separator ‘.’ by a “,” in the MX amount (if decimal separator “.” is present in the source MX amount) or by adding a “,” to the end of the MX amount (if no decimal separator “.” is present in the source MX amount).

Insignificant zeroes -if any- are deleted.

**Format**

**MX\_To\_MTCurrencyAmount**(MXAmount ; MTCurrency, MTAmount)

**Input**

MXAmount: amount in the MX message typed *ActiveOrHistoricCurrencyAndAmount* expressed as a positive value (data type does not allow a sign) with maximum 18digits, maximum 5 fraction digits. The decimal separator “.” is optional, integer values are allowed. The currency is embedded as and XML attribute.

**Output**

MTCurrency: currency in the MT message expressed as 3!a

MTAmount: amount in the MT message expressed as 15d (up to 15 characters including the mandatory decimal separator “,” with the integer part containing at least one character)

**Preconditions**

It is assumed that both MT and MX currencies are either active or active and historic. If it is not the case, validation problem will be expected when the target is more constrained.

The same level of validation on MT and MX is requested to check pairs of (currency, amount) in terms of allowed decimals number.

The MX amount must be restricted to 14 meaningful digits.

**Formal description**

/\*Extract the currency\*/

MTCurrency = MXAmount.XMLAttribute(Ccy)

/\*Translation amount. Amount is a local variable\*/

Amount = MXAmount

/\* if needed remove leading and trailing spaces \*/

Amount = **TrimLeft**(Amount, SPACE)

Amount = **TrimRight**(Amount, SPACE)

/\*Replace the decimal separator “.” -if present- by a “,”. If no decimal separator is present, it is added at the end of the amount\*/

**IF** **IsPresentPattern**(Amount, “.”)THEN

Amount = **ReplacePattern**(MXAmount, “.”, “,”)

/\*Delete insignificant zeroes at the left and right\*/

Amount = **TrimLeft**(Amount, “0”)

Amount = **TrimRight**(Amount, “0”)

/\*Check whether the amount has a strictly fractional part (starts with a “,”). If this is the case, a zero is added before the comma as required in MT fields of the decimal format\*/

**IF** **Substring**(Amount, 1, 1) = “,” THEN

MTAmount = **Concatenate**(“0”, Amount)

**ELSE**

MTAmount = Amount

**ENDIF**

**ELSE**

/\*Remove insignificant zeroes -if any- at the left and add a decimal separator at the end to obtain the amount\*/

Amount = **TrimLeft**(Amount, “0”)

MTAmount = **Concatenate**(Amount, “,”)

**ENDIF**

**Example**

MX Source: <Amt Ccy = "USD">1548.</Amt>

MT Translation: MTCurrency = USD

MTAmount = 1548,

MX Source: <Amt Ccy = "USD">1548.5</Amt>

MT Translation: MTCurrency = USD

MTAmount = 1548,5

MX Source: <Amt Ccy = "USD">1548.00</Amt>

MT Translation: MTCurrency = USD

MTAmount = 1548,

MX Source: <Amt Ccy = "USD">.12</Amt>

MT Translation: MTCurrency = USD

MTAmount = 0,12

### 4.3.8 MX\_To\_MTRemittanceInformation

**Name**

MX\_To\_MTRemittanceInformation

**Business description**

This function translates MX remittance information to remittance information in an MT field 70.

The MX message contains several possible sources of remittance information:

* An optional RemittanceInformation component which can be Structured or Unstructured remittance information but both are exclusive
* An optional RemittanceIdentification, part of (max 1 occurrence)

RelatedRemittanceInformation component to be used when reference is to be made to remittance information sent separately from the payment.

RemittanceInformation and RelatedRemittanceInformation are exclusive.

* A mandatory EndToEndIdentification containing a reference to be passed on from the debtor to creditor side of the payment.
* Purpose of the payment
* Ultimate Creditor and Ultimate Debtor are part of the information to be passed on all along the chain, if present. They must be different from Creditor and Debtor.

If present, UltimateDebtor and UltimateCreditor must be identified with a Name and StructuredPostalAddress (Country and TownName if present) or Other Identification. Nevertheless, if BIC is present, it is translated and the other elements in the ultimate party are ignored.

IMPORTANT

The MTRemittanceInformation is built applying **priority** in the information. This means that the information is truncated in most of the cases. Sign “+” is added at the end of the string to indicate that an information is not completed. If a **full data** is not copied because lack of room then the indicator Flag\_MissingInformation is returned with value “True””.

In all cases, UltimateDebtor and UltimateCreditor will have the priority 1 to be copied in the MTField70. They must be translated to ensure an end-to-end transaction .

When the originating message is MX, the MT remittance information is built with the following tags and priority {/ULTB/, /ULTD/, /PURP/,/ROC/, /URI/, /RELID/, /SRI/+}

Where

/ULTB/ is followed by information about the UltimateCreditor BIC or (Name/Country [/TownName]), TownName is optional or (Name/OtherId) or Name alone or OtherId alone. Depending on the information available, that priority will be applied

/ULTD/ is followed by information about the UltimateDebtor BIC or (Name/Country/TownName), TownName is mandatory or (Name/OtherId) or Name alone or OtherId alone. Depending on the information available, that priority will be applied

/PURP/ is the purpose of the payment.

It is translated to remittance information except if Purpose/Propietary has the following pattern : “:26T:” followed by exact 3 alpha-numeric characters, upper case for the alphabetic characters. This pattern will come from a previous translation MT to MX.

/ROC/ is followed by the EndtoEndIdentification.

It is present if

* The unstructured remittance information is absent AND the value of EndToEndIdentification is different from “NOTPROVIDED”
* The unstructured remittance information is present AND does not contain the code word /ROC/ and the value of EndToEndIdentification is different from “NOTPROVIDED”

If /ROC/ is present in Unstructured remittance information, the EndToEndIdentification translation is ignored.

/URI/ followed by the MX unstructured remittance information

/RELID/ followed by 1 identification of the RelatedRemittanceInformation stored outside the message

/SRI/+ means that a structured remittance information is present in the original message but is not translated.

/URI/, /RELID/ and /SRI/+ are exclusive meaning cannot be present together (even not by pair).

IF the unstructured remittance information is the only element present, the code /URI/ is omitted.

This CR has been raised in order to get back the field 70 as it was in the original payment when the payment is originated in MT, ie Like for like translation

**Format**

**MX\_To\_MTRemittanceInformation**(MXEndToEndId, MXUltimateCreditor, MXUltimateDebtor, MXRemittanceId,

MXRemittanceInfo, MXPurpose ; MTRemittanceInfo)

**Input**

MXUltimateCreditor and MXUltimateDebtor typed PartyIdentification135

MXEndToEndId: reference to be passed on from the debtor to creditor side of the payment typed *Max35Text*.

MXRemittanceId: identification of the remittance information typed *Max35Text* part of RelatedRemittanceInformation component used to reference remittance information sent separately from the payment.

MXRemittanceInfo: the entire structure of the MX remittance information typed *RemittanceInformation16* composed of an Unstructured element typed *Max140Text* and a Structured component typed *StructuredRemittanceInformation16*.

MXPurpose typed Purpose2Choice(code, Proprietary max 35 char)

**Output**

MTRemittanceInfo: remittance information in an MT field 70 format of 4\*35x.

**Preconditions**

At least one of the following elements must be present: UltimateDebtor, UltimateCreditor, Purpose, EndToEndIdentification with a value different from “NOTPROVIDED”, Structured or unstructured remittance information, RelatedRemittanceInformation/RemittanceIdentification.

**Formal description**

/\* Local variables :

MTUltimateCreditor, MTUltimateDebtor, MTROC, MTPurpose,MXCountry, MXTownName, MXName, MXUnstructuredRemittance, MXStructuredRemittanceInformation, UDIndicator, UCIndicator,PURPIndicator, ROCIndicator, MXRelId, RemainingLine, MT70FullString RELID1Indicator, ROCUnstructuredIndicator\*/

All local variables typed string are initiated with empty string.

All local indicators are initiated with “False”.

**Case 1**

***RemittanceInformation and RelatedRemittanceInformation are absent.***

Priority 1:

MXUltimateCreditor and MX UltimateDebtor, if present.

/ULTB/ followed by MXUltimateCreditor BIC

OR

Name/Country[/TownName]

OR

/ULTB/Name/OtherId

OR

/ULTB/Name

OR

/ULTB/OtherId

/ULTD/ followed by MX UltimateDebtor BIC

OR

Name/Country/TownName

OR

/ULTD/Name/OtherId

OR

/ULTD/Name

OR

/ULTD/OtherId

Priority 2

/PURP/ followed by the Purpose of the payment (code or Proprietary).

Priority 3

/ROC/ followed byMX EndToEndIdentification if value is NOT “NOTPROVIDED” and if not present in Unstructured Remittance Information.

**PART 1**

**/\* Fill in information from Ultimate Parties \* /**

UCIndicator = “false”

UDIndicator = “false”

**IF** MXUltimateCreditor **IsPresent**  THEN

**MX\_To\_MTUltimateParty**(MXUltimateCreditor, MTUltimateCreditor)

**IF** **Length**(MTUltimateCreditor) > 0 THEN

UCIndicator = “True”

MTUltimateCreditor = **Concatenate**(“”/ULTB/”, MTUltimateCreditor)

**IF** **Length** (MTUltimateCreditor) > 140 THEN

MT70FullString = **Concatenate**(**Substring**(MTUltimateCreditor,1, 139),”+”))

**ELSE**

MT70FullString = **Substring**(MTUltimateCreditor,1, **Length**(MTUltimateCreditor))

**ENDIF**

**ENDIF** /\* IF Length(MTULtimateCreditor) > 0 \*/

**ENDIF**  /\* MXUltimateCreditor IsPresent \*/

**IF** MXUltimateDebtor **IsPresent** THEN

**MX\_To\_MTUltimateParty**(MXUltimateDebtor, MTUltimateDebtor)

**IF** **Length**(MTUltimateDebtor) > 0 THEN

**{** UDIndicator = “True”

**CASE UCIndicator = “True”**

/\* Next Concatenation starting with “///ULTD/”, \*/

RemaingLine = 140 – **Length**(MT70FullString) - 8

**IF** RemainingLine > 0 THEN

**IF Length**(MTUltimateDebtor) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString, “///ULTD/”,**Substring**(MTUltimateDebtor, 1, RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(MT70FullString, “///ULTD/”, **Substring**(MTUltimateDebtor, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**END CASE** /\* **UCIndicator = “True” \*/**

**CASE UCIndicator = “False ”**

/\* Next Concatenation starting with “/ULTD/”, \*/

RemaingLine = 140 – **Length**(MT70FullString) - 6

**IF** RemainingLine > 0 THEN

**IF Length**(MTUltimateDebtor) > RemainingLine THEN

MT70FullString = **Concatenate**( “/ULTD/”,**Substring**(MTUltimateDebtor, 1, RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(“/ULTD/”, **Substring**(MTUltimateDebtor, 1))

**ENDIF**

**ELSE /\*** RemainingLine = 0 \*/

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

}

**ENDIF** /\* LENGTH(MTUltimateDebtor) > 0 \*/

**ENDIF** /\* MXUltimateDebtor IsPresent \*/

**/\* Add Payment Purpose if there is still room left \*/**

**IF** MXPurpose **NOT IsEmpty** THEN

**IF** MXPurpose/Proprietary **NOT IsEmpty** AND MXPurpose/Proprietary **Has a pattern** “:26T:[A-Z0-9]{3}” THEN

/\* Do not enter the DO section below. No Purpose to translate in RemittanceInformation. This is done elsewhere \*/

**ELSE**

**DO {**

PURPIndicator = “True”

MTPurpose = MXPurpose

**IF** UCIndicator = “True” Or UDIndicator = “True” THEN

/\* Concatenation must start with “///PURP/” \*/

RemaingLine = 140 – **Length**(MT70FullString) - 8

**IF** RemainingLine > 0 THEN

**IF** LENGTH (MTPurpose) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString, “///PURP/”, **Substring**(MTPurpose, 1, RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(MT70FullString ,“///PURP/”, **Substring**(MTPurpose, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ELSE**

/\*UCIndicator = “False” AND UDIndicator = “False”

/\* Concatenation must start with “/PURP/” \*/

RemaingLine = 140 – **Length**(MT70FullString) - 6

**IF** RemainingLine > 0 THEN

**IF** **Length**(MTPurpose) > RemainingLine THEN

MT70FullString = **Concatenate**( “/PURP/”, **Substring**(MTPurpose, 1, RemainingLine-1), “+”))

**ELSE**

MT70FullString = **Concatenate**( “/PURP/”, **Substring**(MTPurpose, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF** /\* IFRemainingLine > 0 \*/

**ENDIF** /\* ENDIF UCIndicator = “True” Or UDIndicator = “True”\*/

**} /\* END DO \*/**

**ENDIF**

**ENDIF**

**/\* Add MXEndToEndIdentification if there is still some room left and if value is not “NOTPROVIDED” \*/**

/\* For developers. The code below handles the case where unstructured remittance info is present in order to define at one place the subfunction for /ROC/ translation. See Case 3 below \*/

/\* Check if “/ROC/ is present in the Unstructured remittance information. If yes, then EndToEndIdentification is not translated \*/

/\*Define a pattern to try to identify that /ROC/ carries meaningful information and is not just the codeword followed by another code word from MT \*/

ROCPattern= /ROC/Text

Where Text cannot start with “//” which is the indication that a new code word starts and must contain at least one character

**IF** **Length**(MXRemittanceInfo.Unstructured) > 0 AND **IsPresentPattern**(MXRemittanceInfo.Unstructured, “ROCPattern”) THEN

ROCUnstructuredIndicator = True

/\* No Translation of EndToEndIdentifcation \*/

**ELSE**

ROCUnstructuredIndicator = False

**ENDIF**

**IF** MXEndToEndIdentification **NOT Equal** to “NOTPROVIDED” AND ROCUnstructuredIndicator = False THEN

MTROC = MXEndToEndIdentification

ROCIndicator = “True”

**IF** UCIndicator = “True” OR UDIndicator = “True” OR PurposeIndicator = “True” THEN

/\* Concatenation must start with “///ROC/” \*/

RemaingLine = 140 – LENGTH(MT70FullString) - 7

**IF** RemainingLine > 0 THEN

**IF** **Length**(MTROC) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString, “///ROC/”, **Substring**(MTROC, 1, RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(MT70FullString ,“///ROC/”, **Substring**(MTROC, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ELSE**

/\* Concatenation must start with “/ROC/” \*/

RemaingLine = 140 – **Length**(MT70FullString) - 5

**IF** RemainingLine > 0 THEN

**IF** **Length(**MTROC) > RemainingLine THEN

MT70FullString = **Concatenate**( “/ROC/”, **Substring**(MTROC, 1, RemainingLine-1), “+”))

**ELSE**

MT70FullString = **Concatenate**( “/ROC/”, **Substring**(MTROC, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ENDIF**

**ENDIF /\* IF** MXEndToEndIdentification NOT Equal to “NOTPROVIDED” AND ROCUnstructuredIndicator = False \*/

**PART 2**

/\* Copy the MT70FullString in 4\*35 Format \*/

**IF** **Length**(MT70FullString)> 105 THEN

MTRemittanceInfo[1] = **Substring**(MT70FullString, 1, 35)

MTRemittanceInfo[2] = **Concatenate**(CRLF,**Substring**(MT70FullString, 36, 70)

MTRemittanceInfo[3] = **Concatenate**(CRLF,**Substring**(MT70FullString, 71, 105)

MTRemittanceInfo[4] = **Concatenate**(CRLF,**Substring**(MT70FullString, 106)

**ELSEIF** **Length**(MT70FullString)> 70 THEN

MTRemittanceInfo[1] = **Substring(**MT70FullString, 1, 35)

MTRemittanceInfo[2] = **Concatenate**(CRLF,**Substring**(MT70FullString, 36, 70)

MTRemittanceInfo[3] = **Concatenate**(CRLF,Substring(MT70FullString, 71)

**ELSEIF Length**(MT70FullString)> 35 THEN

MTRemittanceInfo[1] = **Substring**(MT70FullString, 1, 35)

MTRemittanceInfo[2] = **Concatenate**(CRLF,Substring(MT70FullString, 36)

**ELSEIF** Length(MT70FullString)<= 35 THEN

MTRemittanceInfo[1] = **Substring**(MT70FullString, 1)

**ENDIF**

**Case 2**

***RelatedRemittanceInformation/RemittanceIdentification is present***

Priority 1:

MXUltimateCreditor and MX UltimateDebtor, if present.

/ULTB/ followed by MXUltimateCreditor Name/Country[/TownName]

/ULTD/ followed by MX UltimateDebtor Name/Country/TownName

Where Name and Country are mandatory for UltimateCreditor/UltimateDebor and TownName is mandatory for UltimateDebtor.

Priority 2

/PURP/ followed by the Purpose of the payment (code or Proprietary).

Priority 3

/ROC/ followed byMX EndToEndIdentification if Value is NOT (“NOTPROVIDED”) and IF /ROC/ is not present in UnstructuredRemittanceInformation

Priority 4

/RELID/RelatedRemittanceInformation/RemittanceIdentification (max 1 occurrence)

**IF** RelatedRemittance[1].RemittanceIdentification **IsPresent** THEN

/\* Apply the same logic as in **Case 1 PART 1** ***RemittanceInformation and RelatedRemittanceInformation are absent.*** If room left, add RelatedRemittanceInformation/RemittanceIdentification. \*/

MXRelId = RelatedRemittanceInformation[1].RemittanceIdentification

RELID1Indicator = “True”

**IF** UCIndicator = “True” OR UDIndicator = “True OR PURPIndicator = “True” OR ROCIndicator = “True”

/\* IF Condition could be replaced by

(IF LENGTH(MT70FullString) > 0) \*/

/\* Concatenation must start with “///RELID/” for RelatedRemittance Identification \*/

RemainingLine = 140 – **Length**(MT70FullString) - 9

**IF** RemainingLine >0 THEN

**IF** **Length**(MXRelId) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString ,“/// RELID /”, **Substring**(MXRelId, 1, RemainingLine-1), “+”))

**ELSE**

MT70FullString = **Concatenate**(MT70FullString ,“/// RELID /”, **Substring**(MXRelId, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ELSE**

/\* Concatenation must start with “/RELID/” for RelatedRemittance Identification \*/

RemainingLine = 140 – **Length**(MT70FullString) - 7

**IF** RemainingLine >0 THEN

**IF** **Length**(MXRelId) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString ,“/RELID /”, **Substring**(MXRelId, 1, RemainingLine-1), “+”))

**ELSE**

MT70FullString = **Concatenate**(MT70FullString ,“/RELID /”, **Substring**(MXRelId, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ENDIF /\*** IF LENGTH(MT70FullString) > 0 \*/

/\* Copy the MT70FullString - Apply the same logic as in **Case 1** **PART 2** ***RemittanceInformation and RelatedRemittanceInformation are absent.***. \*/

**END Case 2** /\* RelatedRemittance.RemittanceIdentification IsPresent \*/

**Case 3**

***UnstructuredRemittanceInformation is present***

Priority 1:

MXUltimateCreditor and MX UltimateDebtor, if present.

/ULTB/ followed by MXUltimateCreditor Name/Country[/TownName]

/ULTD/ followed by MX UltimateDebtor Name/Country/TownName

Where Name and Country are mandatory for UltimateCreditor/UltimateDebor and TownName is mandatory for UltimateDebtor.

Priority 2

/PURP/ followed by the Purpose of the payment (code or Proprietary).

Priority 3

/ROC/ followed byMX EndToEndIdentification if Value is NOT (“NOTPROVIDED”) and if /ROC/ followed by some information is not present in unstructured remittance information.

Priority 4

/URI/ followed by Unstructured Remittance Information

/\* Parameter URICodeWord is defined in order to ease the maintenance and implement the CR where /URI/ is omitted if MT70FullString is empty at this point meaning there is no information in MX Ultimate Debtor, MX UltimateCreditor, MX Purpose. EndToEndID is not copied to MT70FullString. Codeword /URI/ is then omitted .

This CR has been raised in order to get back the field 70 as it was in the original payment when the payment is originated in MT, ie Like for like translation \*/

/\* Local variables:

URICodeWord : string \*/

**IF** MXRemittanceInformation.Unstructured **IsPresent** THEN

MXUnstructuredRemittanceInformation = MXRemittanceInformation.Unstructured

/\*Apply the same logic as in **case 1 PART 1** for **Ultimate parties and Purpose and /ROC/** and copy the full MT70FullString \*/

**IF** **Length**(MT70FullString) > 0

/\* concatenation must start with “///URI/” \*/

RemainingLine = 140 – **Length**(MT70FullString) - 7

**IF** RemainingLine > 0 THEN

**IF** **Length**(MXUnstructuredRemittanceInformation) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString, “///URI/”, **Substring**(MXUnstructuredRemittanceInformation, 1, RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(MT70FullString, “///URI/”, **Substring**(MXUnstructuredRemittanceInformation, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ELSE**

**/\*** MT70FullString is empty meaning there is no information in MX Ultimate Debtor, MX UltimateCreditor, MX Purpose. EndToEndID is not copied to MT70FullString. Codeword /URI/ is omitted \*/

. This CR has been raised in order to get back the field 70 as it was in the original payment when the payment is originated in MT, ie Like for like translation \*/

URICodeWord = “”

RemainingLine = 140 – **Length**(MT70FullString) - Length(URICodeWord)

**IF** RemainingLine > 0 THEN

**IF** **Length**(MXUnstructuredRemittanceInformation) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString, URICodeWord, **Substring**(MXUnstructuredRemittanceInformation, 1, RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(MT70FullString, URICodeWord, **Substring**(MXUnstructuredRemittanceInformation, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ENDIF**

/\* Copy the MT70FullString to MTRemittanceInformation - Apply the same logic as in **Case 1** **PART 2** ***RemittanceInformation and RelatedRemittanceInformation are absent.***. \*/

**ENDIF** /\* MXRemittanceInformation.Unstructured IsPresent \*/

**/\* END Case 3 \*/**

**Case 4**

***StructuredRemittanceInformation is present***

***No information from StructuredRemittanceInformation is copied. Only /”SRI/+ “ is added at the end of the string, if room left to indicate the information is missing.***

Priority 1:

MXUltimateCreditor and MX UltimateDebtor, if present.

/ULTB/ followed by MXUltimateCreditor Name/Country[/TownName]

/ULTD/ followed by MX UltimateDebtor Name/Country/TownName

Where Name and Country are mandatory for ULtimateCreditor/UltimateDebor and TownName is mandatory for UltimateDebtor.

Priority 2

Purpose of the Payment

Priority 3

/ROC/ followed byMX EndToEndIdentification if value is NOT “NOTPROVIDED”

**IF** RemittanceInformation.StructuredRemittanceInformation **IsPresent** THEN

/\* Apply the same logic as in **Case 1** **PART 1** ***RemittanceInformation and RelatedRemittanceInformation are absent. \*/***

*/\* If room left then /SRI/+ is added to indicate that the information is truncated, i.e., Not translated \*/*

**IF** **Length**(MT70FullString) > 0 THEN

/\* concatenation must start with “///SRI/+” \*/

RemaingLine = 140 – **Length**(MT70FullString) - 8

**IF** RemainingLine > 0 THEN

MT70FullString = **Concatenate**(MT70FullString, “///SRI/+”)

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**ELSE**

/\* concatenation must start with “/SRI/+” \*/

MT70FullString = **Concatenate**(MT70FullString, “/SRI/+”)

/\* Copy the MT70FullString to MTRemittanceInformation - Apply the same logic as **in Case 1** **PART 2** ***RemittanceInformation and RelatedRemittanceInformation are absent.***. \*/

**ENDIF**

**ENDIF** /\* RemittanceInformation.StructuredRemittanceInformation \*/

### 4**.3.9 MXText\_To\_MT72**

**Name**

MXText\_To\_MT72

**Business description**

The function transforms an MX text string into an MT text format preceded by a code (passed as input parameter of the function) and splits the outcome into a maximum number of MT field 72 lines of 35x characters (length passed as input parameter of the function). Sign “+” is added at the end of the MT string to indicate that the information is truncated.

**Format**

**MXText\_To\_MX72**(MXText, MTCode, Length ; MT72)

**Input**

MXText: MX string of characters.

MTCode: code as used in an MT field 72 (for example /REC/, /ACC/, /BNF/…). Length: maximum length of the output string of characters.

**Output**

MT72: content of an MT field 72 in lines of maximum 35 characters.

**Preconditions**

None.

**Formal description**

/\*Combine code with MXText and restrict the output to the maximum length as indicated by the input parameter.

/\* Local variables :

CodeText : string

TempLength : integer\*/

CodeText = **Concatenate**(MTCode, MXText)

TempLength = **Length**(CodeText)

**IF** TempLength > Length THEN

CodeText = **Concatenate**(**Substring**(CodeText,1, TempLength – 1),”+”)

**ELSE**

CodeText = Concatenate(MTCode, MXText)

**ENDIF**

/\*Field 72 is defined by 6 lines of 35 characters with “//” as line continuation pattern and Carriage Return Line Feed (CRLF) as MT line separator between consecutive lines. The function below brings the output string in line with this format\*/

MT72 = **SplitInLines**(CodeText, 35, “//”)

**Example MX Source (InstructionForCreditorAgent used without code):**

<InstrForCdtrAgt>

<InstrInf>Please process according to service level agreement AD1.1</InstrInf>

</InstrForCdtrAgt>

**MT Translation (input code /ACC/ and length restriction of 63 characters):**

:72:/ACC/Please process according to se

//rvice level agreement AD1.1

### 4**.3.10 MX\_To\_MT72FullField**

**Name**

MX\_To\_MT72 FullField

**Business Description**

The function combines elements from the MX message that must be carried to the next agent in order to allow the processing of the payment all along the chain.

Depending on the room left and the presence of the elements in the physical message, the following priority will be applied with possible missing or truncated information:

Priority 1 : IntermediaryAgent2,3 (/INTA/)\*

Priority 2 : PaymentTypeInformation/ServiceLevel (/SVCLVL/) \*(excluded code SDVA or with a pattern “G00n” where n is an integer

Priority 3 : PaymentTypeInformation/LocalInstrument (/LOCINS/)\* (excluded code CRED, CRTS, SPAY, SPRI, SSTD)

Priority 4 : PaymentTypeInformation/CategoryPurpose (/CATPURP/)\* (excluded code INTC and CORT, Proprietary <> “INTC CORT” string

Priority 5 : InstructionForCreditorAgent (excluded code 23E like /HOLD/, /CHQB/, /PHOB/, /TELB/ translated by using MX\_To\_MT23E)

Priority 6: InstructionForNextAgent (excluded “/FIN53/BIC” structure)

Priority 7: PreviousInstructingAgent1,2,3 (/INS/)

\*means new code words to be used in Field72

If a data is truncated (ie., partially present), “+” is added at the end of the string;

If an information cannot be copied in field 72 (eg., PreviousIntructingAgent2,3), an indicator Flag\_MissingInformation is returned with value “True”.

There is a special request from Japan community to use the CreditorAgent/Branch id/Identification for domestic payments and to copy it to Field72 of MT103 with "/ACC/".

CreditorAgent/Branch is translated to Field 72 only if the CreditorAgent’s BICs or PostalAddress/Country (value “JP”) is used otherwise there is no way to identify it is a domestic Japan payment

This element will be treated just before the information from InstructionForCreditorAgent.

The function can be used to convert for the first time an MX message created in MX to an MT message or to convert a MX message already converted previously in MT in a sequence like MX to MT to MX or to convert a MX message initiated in MT.

**IntermediaryAgent2,3** can only be present if the original message is a MX message. That information must be carried to continue the payment process and a new code will be used /INTA/ in order to fit with field 72 structure. This code can be repeated max 2 times. It can be understood as an instruction for IntermediaryAgent1 (/INT/) on how to continue the chain but in order to indicate it is a new information coming from an original MX message, a new code word is used.

IntermediaryAgent2,3 must be identified at a minimum with a BIC OR (Name [and TownName and Country])for cross border transactions or by ClearingSystemMemberIdentification for domestic transactions.

BIC is the preferred option for the translation.

The ISO Clearing System Identification is not converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

Proposal MT72 field template for IntermediaryAgent2,3:

/INTA/BIC OR /INTA/Name [**(**Country**(**TownName] OR /INTA/ClearingSystemMemberIdentification

Where “(“ is used as a separator easy to identify in a string by searching a pattern “(“ followed by exactly 2 alphabetic characters which must be an ISO country code followed by “(“.

Country and TownName will be provided if present in the MX

If IntermediaryAgent2,3 are present, with the above information, 6 lines might be consumed and no room left for other information as shown below, if present in MX.

ServiceLevel (ISO code or Proprietary) is translated to Field 72 with code word /SVCLVL/. For the ISO codes, only codes different from “SDVA” and having not a pattern like “G00n” where n is an integer, are translated to field 72.

“SDVA” is translated to 23E. gpi Codes (G00n) are translated to FIN Block3/EndToEndReference/ServiceTypeIdentifier in a different function. Up to 3 occurrences /SVCLVL/ are possible, although it is more likely that maximum 2 will be translated to field 72.

LocalInstrument (ISO code or Proprietary) is translated to Field 72 with code word /LOCINS/. If Proprietary is present and has a value CRED, CRTS, SPAY, SPRI or SSTD, then fixed default value “CRED” is translated to 23B. Only one occurrence of Local Instrument is allowed.

CategoryPurpose (ISO code or Proprietary) is translated to Field 72 with code word /CATPURP/ if ISO code is not INTC or CORT or Proprietary does not contain the exact pattern “INTC CORT” coming from a previous MT to MX translation. Only one occurrence is allowed.

InstructionForCreditorAgent, when InstructionInformation is used with a code, both information will be mapped to field 23E in MT (except if the code is “CHQB” for which no additional information can be provided in MT23E). Another function is used.

When InstructionInformation is used with no code or with code “CHQB” , then it will be translated to MT field 72 with code /ACC/.

InstructionForNextAgent.InstructionInformation may be present without code in case the information is originated from an MX message or could contain codes like /REC/, /INT/ or /ProprietaryCode/ if the element results from a translation MT to MX or if the user replicates in MX the MT practices from field 72. When InstructionInformation is originated from MX without /Code/, it will be translated with the code /REC/ to MT.

**The codes /FIN53/** followed by BIC can be found in InstructionForNextAgent.InstructionInformation from a previous translation MT to MX. Although they should have been be removed from the receiver side when the next leg of the payment was created unless the same code is used in MX to mimic in MX the specific MT scenario where 53A/BIC is translated to InstructionForNextAgent/FIN53A/ (scenario described in METAFCT001 in MT103 to pacs.008).There is no check on the BIC’s value following the code. But this is not encouraged as it is a misuse of the standards. Code /FIN53/ and the BIC information is translated back to field 53A by another function. Refer also to SubfunctionInstructionForNextAgent for more information on the extraction of /FIN53/ related information.

PreviousInstructingAgent1,2,3 exist already in MT and are transported with the code /INS/followed by BIC or Name in field 72 or they could be defined for the first time in an original MX message. From an original MX message, if the postal address is present, it must be a structured postal address while if the presence of these agents in MX is already resulting from an MT to MX translation, only a unstructured postal address (eg., AddressLine) is present with value “NOTPROVIDED”. Nevertheless the translation MX to MT will ignore the PostalAddress as this information is currently not available in MT. BIC is the preferred option for the translation.

In case both BIC and Name are absent in MX, the ClearingSystemMemberIdentification will be translated in order to cater for domestic transactions which might only use the ClearingSystemMemberIdentification. This will only happen if the original message is MX.

The ISO Clearing System Identification is not converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

**Format**

MX\_To\_MT72 FullField

**Input** MXIntermediaryAgent2 , MXIntermediaryAgent3, MXPreviousInstructingAgent1, MXPreviousInstructingAgent2, MXPreviousInstructingAgent3, InstructionForCreditorAgent, InstructionForNextAgent, CategoryPurpose, ServiceLevel, LocalInstrument, MXCreditorAgent, MXSettlementMethod, BusinessApplicationHeader

**Output**

MT72: content of an MT field 72 in lines of maximum 6 \* 35 characters.

**Preconditions**

At least one of the input parameters is present (eg., not empty)

**Formal description**

/\* Local variables :

MXAgent typed BranchAndFinancialInstitutionIdentification6, MTAgent is a string MTTemp72 has a structure 3\*35 char, PresentInfo and IsCategoryPurposePresent are booleans, MXInstruction is a string, MTInstruction is a string, NumberofEmptyLines is a integer , MTInstructionLength is integer, NumberOf char is integer, MTNoCodeInstruction is a string\*/

**/\* Manage the priorities and depending on the room left additional information is translated to Field 72\*/**

**/\* Priorities and Subfunction call section \*/**

**IF** (intermediaryAgent2) **IsPresent** THEN

**Call** **SubfunctionIntermediaryAgents**

/\*For developers only : Subfunction described in MX\_To\_MT72FullField2 but replace

“Exit Function MX\_To\_MT72FullField2” by Exit Function “MX\_To\_MT72FullField” \*/

**ENDIF**

**IF** (PaymentTypeInformation/ServiceLevel) **IsPresent**

THEN

**Call** **SubfunctionServiceLevel2**

/\*For developers only – Subfunction described below, is different from MX\_To\_MT72FullField2; Multiple occurrences allowed. Some values of Code handled by another function \*/

**ENDIF**

**IF** [PaymentTypeInformation/LocalInstrument/Code **IsPresent** OR (PaymentTypeInformation/LocalInstrument/Proprietary **IsPresent** AND **NOT InList**{CRED,CRTS,SPAY,SPRI, SSTD} ) ] THEN

**Call** **SubfunctionLocalInstrument**

/\*for developers only - Subfunction described in MX\_To\_MT72FullField2 but to replace “Exit Function MX\_To\_MT72FullField2 “by Exit Function MX\_To\_MT72FullField \*/

**ENDIF**

**IF** (PaymentTypeInformation/CategoryPurpose/Code **IsPresent** AND Code **NOT InLIST** {INTC, CORT}) OR

(PaymentTypeInformation/CategoryPurpose/Proprietary **IsPresent** AND PaymentTypeInformation/CategoryPurpose/Proprietary **NOT Equal** to “INTC CORT” (ie “INTC”SPACE”CORT”)

THEN

**Call** **SubfunctionCategoryPurpose**

/\*for developers only - Subfunction described in MX\_To\_MT72FullField2 but to replace “**Exit Function MX\_To\_MT72FullField2 “by Exit Function MX\_To\_MT72FullField** \*/

**ENDIF**

**IF** **IsPresent**(InstructionForCreditorAgent OR CreditorAgent/BranchIdentification) THEN

**Call** **SubfunctionInstructionforCreditorAgentAndJP**

/\*for developers, not the same function as in MX\_To\_MT72FullField2\*/

**ENDIF**

**IF** InstructionForNextAgent **IsPresent** THEN

**Call** **SubfunctionInstructionForNextAgent**

/\*for developers only - Subfunction described in MX\_To\_MT72FullField2 but to replace “**Exit Function MX\_To\_MT72FullField2 “by Exit Function MX\_To\_MT72FullField** \*/

**ENDIF**

**IF** PreviousInstructingAgent1 **IsPresent** THEN

**Call** **SubfunctionPreviousInstructingAgent**

/\*for developers only - Subfunction described in MX\_To\_MT72FullField2 but to replace “**Exit Function MX\_To\_MT72FullField2 “by Exit Function MX\_To\_MT72FullField** \*/

**ENDIF**

**/\* Subfunction definition part \*/**

**SubfunctionServiceLevel2**

/\* LocalVariables

MXServiceLevel :string

\*/

**For n**= 1 to 3

**IF** CreditTransferTransactionInformation/PaymentTypeInformation/ServiceLevel[n]/Proprietary **IsPresent** OR (CreditTransferTransactionInformation/PaymentTypeInformation/ServiceLevel[n]/Code **IsPresent** AND Code **NOT InList** {SDVA} AND Code **has a pattern <>** “G00n, where n is integer) THEN

**{**

MXServiceLevel = ServiceLevel[n]

/\* either code or proprietary \*/

**IF** **Length**(MXServiceLevel) > 0 THEN

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField

**ENDIF**

MXServiceLevel = **Concatenate** (“/SVCLVL/”, MXServiceLevel)

/\* internal only for developers , save the code “/SVCLVL/” in a variable in order to cater for value change later and change to be done at one place\*/

NumberOfEmptyLines = 6 – ReturnFirstLineEmpty (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MXServiceLevel, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** LENGTH(MXServiceLevel) > 0 \*/

**}**

**ENDIF** /\* ServiceLevel[n] isPresent \*/

**Next n**

**SubfunctionInstructionForCreditorAgentAndJP**

**/\* Extract InstructionForCreditorAgent and CreditorAgent/BranchIdentification ONLY for Japan domestic payments \*/**

/\* The subfunction extracts information from InstructionForCreditorAgent.InstructionInformation and translates to field 72 /ACC/ all information that is not translated to 23E.

This may come from different scenarios:

-Textual information linked to code /CHQB/ (/CHQB/ being present in Code or InstructionInformation)

-Textual information present and not related to any code in the List {HOLD, PHOB, TELB}

In the string to analyse, all codes in list {HOLD, PHOB, TELB} and related information are deleted as already translated to 23E.

The function also deletes /CHQB/ but not the related information because in 23E, no information is allowed with code CHQB. So if there is information following /CHQB/ it is translated with /ACC/

All the information remaining after clearing InstructionInformation (ie removing codes and related information translated to 23E) or information linked to code “/CHQB/” are translated to Field 72 /ACC/ by concatenation. This allows to avoid loosing information but there is no quarantee about the order. When several text pieces are concatenated, they are separated by a space.

As this function is complementary to MX\_To\_MT23E and MX\_To\_MT23E\_BIS, the same concatenation of the 2 occurrences as described in MX\_To\_MT23E\_BIS is applied.

/\* LocalVariables:

MXJPInstruction, MXInstruction: string

CodeTable[] : table of codes

m:integer

CodeList : list of codes {/HOLD/, /PHOB/, /TELB/, /CHQB/}

ACCTable[] : table of string

\*/

CodeTable[1] = “/HOLD/”

CodeTable[2] = “/CHQB/”

CodeTable[3] = “/PHOB/”

CodeTable[4] = “/TELB/”

CodeTable[5] = “/TempAcc/”

**IF** **IsPresent**(MXCreditorAgent.BranchIdentification.Identification) AND (**IsPresent**(MXCreditorAgent.FinancialInstitutionIdentification.BICFI OR **IsPresent**(MXCreditorAgent.FinancialInstitutionIdentification.PostalAddress.Country)) THEN

**IF Substring**(MXCreditorAgent.FinancialInstitutionIdentification.BICFI, 5,2) = “JP” OR

            MXCreditorAgent.FinancialInstitutionIdentification.PostalAddress.Country = “JP”

THEN

MXJPInstruction = MXCreditorAgent.BranchIdentification.Identification

**ENDIF**

**ENDIF**

/\* Define the string to analyse \*/

**IF IsPresent**(InstructionForCreditorAgent[1] AND **IsAbsent**(InstructionForCreditorAgent[1].Code) AND

**IsPresent**(InstructionForCreditorAgent[2] AND **IsAbsent**(InstructionForCreditorAgent[2].Code)

/\* 2 occurrences of InstructionInformation are concatenated \*/

**THEN Call SubfunctionAnalyse2OccurrencesNoCode**

/\* The subfunction fills ACCTable[] containing all the textual information to be translated to field 72 /ACC/. Subfunction described below \*/

**ElSE**

**Call SubfunctionAnalysePer1Occurrence**

/\* Each occurrence of InstructionInformation is analysed separately in thesubfunction \*/

/\* The subfunction fills ACCTable[] containing all the textual information to be translated to field 72 /ACC/. Subfunction described below \*/

**ENDIF**

/\* Build the string out of the ACCTable \*/

**IF NumberOfOccurrences**(ACCTable[])> 0 Then

MXInstruction = ACCTable[1]

**For** m = 2to **NumberOfOccurrences**(ACCTable[])

MXInstruction = **Concatenate**(MXInstruction, SPACE, ACCTable[m])

Next m

**ENDIF**

/\* Concatenate with MXJPInstruction \*/

**IF** **Length**(MXInstruction) > 0 THEN

**IF** (MXJPInstruction) > 0 THEN

MXInstruction= **Concatenate**(MXJPInstruction,SPACE,MXInstruction)

**ENDIF**

**ELSE**

**IF** **Length**(MXJPInstruction) > 0 THEN

MXInstruction = MXJPInstruction

**ENDIF**

**ENDIF**

**IF** MXInstruction NOT IsEmpty THEN

MTInstruction = **Concatenate**(“/ACC/”, MXInstruction)

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField

**ENDIF**

NumberofEmptyLines = 6 - **ReturnFirstLineEmpty** (MT72, 6)+1

/\* Format information to build field 72 structure with a “+” sign if information is truncated \*/

/\* Append to MT72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTInstruction ,MT72; MT72)

**ENDIF** /\* MXInstruction NOT IsEmpty \*/

**SubfunctionAnalyse2OccurrencesNoCode**

/\* Local variables

MXInstruction, InstructionCode : string

\*/

/\* Build the string to analyse \*/

{**IF Length**(InstructionForCreditorAgent[1].InstructionInformation)<140 **AND NOT WithinList**(**Substring**(InstructionForCreditorAgent[2].InstructionInformation),1,6), CodeList)THEN

MXInstruction = **Concatenate**(InstructionForCreditorAgent[1].InstructionInformation,”/TempACC/”, InstructionForCreditorAgent[2].InstructionInformation)

**ELSE**

MXInstruction = **Concatenate**(InstructionForCreditorAgent[1].InstructionInformation, InstructionForCreditorAgent[2].InstructionInformation)

**ENDIF**}

InstructionCode = “”

**call SubfunctionExtractACC**(InstructionCode,CodeTable[], MXInstruction; ACCTable[])

**End SubfunctionAnalyse2OccurrencesNoCode**

**SubfunctionAnalysePer1Occurrence**

/\* Each occurrence of InstructionForCreditorAgent is analysed separately \*/

/\* local Variables

j,k,m,t:integer

MXInstruction, InstructionCode : string

ACCTable1[], ACCTable2[] : table of string \*/

**For j** = 1 to NumberOfOccurences(InstructionForCreditorAgent)

**IF IsPresent**(InstructionForCreditorAgent[j].InstructionInformation THEN

/\* IF InstructionInformation is absent, then there is nothing to translate to field 72 /ACC/ \*/

MXInstruction = InstructionForCreditorAgent[j].InstructionInformation

**IF IsPresent**(InstructionForCreditorAgent[j].Code) THEN

InstructionCode = InstructionForCreditorAgent[j].Code

**ELSE**

InstructionCode = “”

**ENDIF**

**Call SubfunctionExtractACC**(InstructionCode,CodeTable[], MXInstruction; ACCTable[])

/\* TempACC is not used when 2 occurrences are analysed separately, so will not be found in the string to analyse \*/

**IF** j = 1 Then

**For k** = 1 to NumberOf Occurrences(ACCTable[])

ACCTable1[k] = ACCTable[k]

**Next** k

**ELSE**

**For** k = 1 to NumberOf Occurrences(ACCTable[])

ACCTable2[k] = ACCTable[k]

**Next k**

**ENDIF**

**ENDIF**

**Next j**

/\* Merge the 2 tables \*/

j = NumberOfOccurrences(ACCTable1[])

k = NumberOfOccurrences(ACCTable2[])

m = NumberOfOccurrences(ACCTable[])

/\* Reset ACCTable[] \*/

**IF** m > 0 THEN

**For** t = 1 to m

ACCTable[t] = “”

**Next t**

**ENDIF**

**IF** j > 0 Then

**For** t = 1 to j

ACCTable[t] = ACCTable1[t]

**Next t**

**ENDIF**

**IF** k > 0 THEN

**For** t = 1 to k

j = j+1

ACCTable[j] = ACCTable2[t]

**Next** t

**ENDIF**

/\* ACCTable contains all the information to be translated to Field 72 /ACC/ \*/

/\* End **SubfunctionAnalysePer1Occurrence** \*/

**SubfunctionExtractACC**(InstructionCode,CodeTable[], MXInstruction; ACCTable[])

/\* local variables

k,j,t:integer

MXText,Temp,RemainingString, FoundPatternText : string

Table[] is made of Table.Code and Table.Text

\*/

j = 0

RemainingString = MXInstruction

**For k** = 1 to 5

/\* k counts the number of permutations\*/

**IF Length**(RemainingString) > 0 THEN

**IF IsPresentPattern**(RemainingString, CodeTable[1]) THEN

MXText = **ExtractBetweenPattern**(RemainingString, CodeTable[1],{CodeTable[2], CodeTable[3], CodeTable[4], CodeTable[5]})

j= j + 1

Table[j].Code = CodeTable[1]

Table[j].Text = MXText

FoundPatternText = **Concatenate**(CodeTable[1], MXText)

RemainingString = **DeletePattern**(RemainingString, FoundPatternText)

**ENDIF**

/\* Permutation of the codes \*/

Temp = Code[1]

For t = 2 to 5

CodeTable[t-1] = CodeTable[t]

Next t

Code[5] = Temp

**ENDIF**

**Next** k

/\* Just keep information, not translated to 23E ie linked to “/TempACC/” or “/CHQB/” \*/

**IF** j > 0 THEN

t = 0

**For** k = 1 to j

**IF** Table[k].Code = “/TempACC/” or “/CHQB/” THEN

**IF Length**(Table[k]. Text) > 0 THEN

t = t + 1

ACCTable[t] = Table[k]. Text

**ENDIF**

**ENDIF**

**Next** k

**ENDIF**

**IF Length**(RemainingString)> 0 AND (InstructionCode = “” OR “CHQB”) THEN

t = t+1

ACCTable[t] = RemaingString

**ENDIF**

**/\* End SubfunctionExtractACC \*/**

#### 

### 4.3.11 AppendComplexMT72

**Name**

AppendComplexMT72

**Business description**

The function adds a string made of one “/Code/” followed by Information to a Field 72 taking into account the number of remaining lines available. The information is truncated with “+” sign if all the information cannot be copied.

**Format**

**AppendComplexMT72**(NumberOfEmptyLines, MTInstruction, MT72; MT72)

**Input**

NumberOfEmptyLines: integer; MTInstruction : string, MT72 : string.

**Output**

MT72 : string

**Preconditions** None.

**Formal description**

/\*Local variables:

MTInstuctionLength: integer

MTTemp72 : string \*/

/\* Format information to build field 72 structure with a “+” sign if information is truncated \*/

MTInstuctionLength = **Length**(MTInstruction)

**IF** MTInstructionLength < 36 THEN

**AppendToNextLine**(MTInstruction, MT72)

**ELSEIF** MTInstructionLength < 69THEN

/\* 2 lines needed \*/

**IF** NumberOfEmptyLines > 1 THEN

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSE** /\* 1 line is available \*/

**AppendToNextLine**(**Concatenate**(**Substring**(MTInstruction, 1,34), “+”), MT72)

**ENDIF**

**ELSEIF** MTInstructionLength < 102 THEN

/\* 3 lines needed \*/

**IF** NumberOfEmptyLines > 2 THEN

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSEIF** NumberOfEmptyLines > 1 THEN

/\* 2 lines available \*/

MTInstruction = **Concatenation** (**Substring**(MTInstruction, 1, 67), “+”)

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSE** /\* 1 line is available \*/

**AppendToNextLine**(**Concatenate**(**Substring**(MTInstruction, 1,34), “+”), MT72)

**ENDIF**

**ELSEIF** MTInstructionLength < 135 THEN

/\* 4 lines needed \*/

**IF** NumberOfEmptyLines > 3 THEN

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTtemp72, MT72)

**ELSEIF** 1< NumberOfEmptyLines <= 3 THEN

/\* 2 or 3 lines available \*/

NumberOf Char = NumberOfEmptyLines \* 35 – (NumberOfEmptyLines-1)\*2

MTInstruction = **Concatenation** (**Substring**(MTInstruction, 1, NumberOf Char-1), “+”)

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSE** /\* 1 line is available \*/

**AppendToNextLine**(**Concatenate**(**Substring**(MTInstruction, 1,34), “+”), MT72)

**ENDIF**

**ELSEIF** MTInstructionLength < 168 THEN

/\* 5 lines needed \*/

**IF** NumberOfEmptyLines > 4 THEN

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSEIF**  1< NumberOfEmptyLines <= 4 THEN

/\* 2, 3 or 4 lines available \*/

NumberOf Char = NumberOfEmptyLines \* 35 – (NumberOfEmptyLines-1)\*2

MTInstruction = **Concatenation** (**Substring**(MTInstruction, 1, NumberOf Char-1), “+”)

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSE** /\* 1 line is available \*/

**AppendToNextLine**(**Concatenate**(**Substring**(MTInstruction, 1,34), “+”), MT72)

**ENDIF**

**ELSEIF** MTInstructionLength < 201 THEN

/\* 6 lines needed \*/

**IF** NumberOfEmptyLines > 5 THEN

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSEIF** 1< NumberOfEmptyLines <= 5 THEN

/\* 2, 3 , 4 or 5 lines available \*/

NumberOf Char = NumberOfEmptyLines \* 35 – (NumberOfEmptyLines-1)\*2

MTInstruction = **Concatenation**(**Substring**(MTInstruction, 1, NumberOf Char-1), “+”)

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ELSE**  /\* 1 line is available \*/

**AppendToNextLine**(**Concatenate**(Substring(MTInstruction, 1,34), “+”), MT72)

**ENDIF**

**ELSE**

**/\* Truncation is needed as MTInstruction length >= 201 \*/**

NumberOf Char = NumberOfEmptyLines \* 35 – (NumberOfEmptyLines-1)\*2

MTInstruction = **Concatenation** (**Substring**(MTInstruction, 1, NumberOf Char-1), “+”)

MTTemp72 = **SplitInLines**(MTInstruction,35, “//”)

**AppendToNextLine**(MTTemp72, MT72)

**ENDIF**

### 4.3.12 MX\_To\_MT23E

**Name**

MX\_To\_MT23E

**Business description**

The function extracts the code CHQB, HOLD, PHOB, TELB scanning the 2 occurrences of InstructionForCreditorAgent[n].Code and translates them to 23E with related InstructionInformation. The function also checks if the element InstructionInformation[n] carries one or several of the codes above. This could result from a previous translation MT to MX (see function MT\_To\_MXInstructionForCreditorAgent). Each occurrence of InstructionInformation is analysed separately (ie., no concatenation) due to the translation MT to MX.

IF InstructionForCreditorAgent[n].Code is present and code /CHQB/ is present in InstructionForCreditorAgent[n].InstructionInformation follows by textual information, the textual information is translated in *SubfunctionInstructionForCreditorAgentAndJP called from* MX\_To\_MT72FullField. As explained this pattern is possible from a previous translation MT to MX but /CHQB/**Text** is not expected from MT in MXInstructionInformation.

Similarly, if the code in InstructionForCreditorAgent[n].Code is /CHQB/ and the string in InstructionInformation[n] is not empty after having removed the possible codes and related information, the remaining string is translated to field 72 /ACC/ in the same subfunction *SubfunctionInstructionForCreditorAgentAndJP.*

IF the code in InstructionForCreditorAgent[n].Code is not /CHQB/, the remaining string is attached to the code when translated to 23E.

Information related to a Code is truncated if length is greater that 30 characters and truncation is indicated with a “+” sign.

Example

<InstrForCdtrAgt>

<Cd>TELB</Cd>

<InstrInf>Text1/CHQB/Text2</InstrInf>

</InstrForCdtrAgt>

Translation:

:23E:CHQB

:23E:TELB/Text1

A similar function MX\_To\_MT23E\_BIS is used when InstructionForCreditorAgent.Code is absent to extract possible codes {/CHQB/, /HOLD/, /PHOB/, /TELB/} from InstructionInformation.

Limitation of the function: if InstructionForCreditorAgent[n].Code is present, only the

InstructionForCreditorAgent[n].InstructionInformation after having removed the possible codes in it, is attached to the Code, ie., not the information from the other occurrence of InstructionInformation. So if a code is split between the 2 occurrences of InstructionInformation, it will not be identified as a code and will be translated as part of the occurrence where it occurs (at least if no truncation is applied).

For example

Occurrence 1 of InstructionForCreditorAgent:

HOLD

Text1…. /PH (all 140 char are used)

Occurrence 2 of InstructionForCreditorAgent:

No code

OB/Text2

:23E:HOLD/Text1 (truncated after 29 char and “+)

:72:/ACC/OB/Text2

The second occurrence is translated by the function MX\_To\_MT23E\_BIS

**Format**

**MX\_To\_MT23E**(InstructionForCreditorAgent[1], InstructionForCreditorAgent[2]; 23E[n])

**Input**

InstructionForCreditorAgent typed InstructionForCreditorAgent1, max 2 occurrences

**Output**

Field23E (multiple occurrences if any)

**Preconditions**

InstructionForCreditorAgent.Code IsInList {CHQB, HOLD, PHOB, TELB}

**Formal description**

/\* Local Variables :

Instruction[4] : structure Field 23E

MXText, RemainingString ,FoundPatternText : string

j is integer \*/

j = 0

**For n** = 1 to 2 /\*Max 2 occurrences of InstructionForCreditorAgent \*/

**IF** InstructionForCreditorAgent[n].Code **IsInList** {CHQB, HOLD, PHOB, TELB} THEN

/\* search for other codes and related instruction in InstructionForCreditorAgent[n].InstructionInformation \*/

RemainingString = InstructionForCreditorAgent[n].InstructionInformation

**IF IsPresentPattern**(InstructionForCreditorAgent[n].InstructionInformation,”/CHQB/”) THEN

j = j +1

Instruction[j].Instruction Code =”CHQB”

MXText = **ExtractBetweenPattern**( InstructionForCreditorAgent[n].InstructionInformation, "/CHQB/",{"/HOLD/","/PHOB/","/TELB/"}

/\* no additional information is expected with code “CHQB”, at least if the payment is originated in MT. If found, ignore it in this function. It is translated to Field 72 with code /ACC/ from SubfunctionInstructionforCreditorAgentAndJP called in MX\_To\_MT72FullField \*/

FoundPatternText = **Concatenate**(“/CHQB/”, MXText)

RemainingString = **DeletePattern**(InstructionForCreditorAgent[n].InstructionInformation, FoundPatternText)

**ENDIF** /\* ENDIF IsPresentPattern “/CHQB/” \*/

/\* Reinitialise MXText for next search \*/

MXText = “”

**IF IsPresentPattern**(InstructionForCreditorAgent[n].InstructionInformation,”/HOLD/”) THEN

j = j +1

Instruction[j].Instruction Code =”HOLD”

MXText = **ExtractBetweenPattern**( RemainingString, "/HOLD/",{"/CHQB/","/PHOB/","/TELB/"}

Instruction[j]. Additional Information = MXText

FoundPatternText = **Concatenate**(“/HOLD/”, MXText)

RemainingString = **DeletePattern**(RemainingString, FoundPatternText)

**ENDIF**  /\* ENDIF IsPresentPattern “/HOLD/” \*/

/\* Reinitialise MXText for next search \*/

MXText = “”

**IF IsPresentPattern**(InstructionForCreditorAgent[n].InstructionInformation,”/PHOB/”) THEN

j = j +1

Instruction[j].Instruction Code =” PHOB”

MXText = **ExtractBetweenPattern**(RemainingString,

"/ PHOB/",{"/CHQB/","/HOLD/","/TELB/"}

Instruction[j]. Additional Information = MXText

FoundPatternText = **Concatenate**(“/PHOB /”, MXText)

RemainingString = **DeletePattern**(RemainingString, FoundPatternText)

**ENDIF**  /\* ENDIF IsPresentPattern “/PHOB/” \*/

/\* Reinitialise MXText for next search \*/

MXText = “”

**IF IsPresentPattern**(InstructionForCreditorAgent[n].InstructionInformation,”/TELB/”) THEN

j = j +1

Instruction[j].Instruction Code =” TELB”

MXText = **ExtractBetweenPattern**(RemainingString,

"/TELB/",{"/CHQB/","/PHOB/","/HOLD/"}

Instruction[j]. Additional Information = MXText

FoundPatternText = **Concatenate**(“/TELB/”, MXText)

RemainingString = **DeletePattern**(RemainingString, FoundPatternText)

**ENDIF**  /\* ENDIF IsPresentPattern “/TELB/” \*/

/\* Store the code \*/

j=j+1

Instruction[j].Instruction Code = InstructionForCreditorAgent[n].Code

/\* If Remaing string is not empty then the information is related to the code in InstructionForCreditorAgent[n].Code \*/

**IF** RemainingString **NOT IsEmpty** THEN

/\*Field 23E with code CHQB does not allow additional information – Information is translated to field 72 using the function MX\_To\_MT72FullField \*/

**IF** InstructionForCreditorAgent[n].Code = “CHQB” THEN

RemainingString = “”

**ENDIF**

Instruction[j]. Additional Information = RemainingString

**ENDIF**

**ENDIF** /\* End IF InstructionForCreditorAgent[n].Code IsInList {CHQB, HOLD, PHOB, TELB} \*/

n=n+1 /\* Next n \*/

/\* sort the codes and their associated additional information table in order to meet MT rules. SDVA, INTC and CORT if already present in 23E, must be before HOLD, CHQB, PHOB, TELB. Sequence in 23 E is therefore SDVA, INTC, CORT, HOLD, CHQB, PHOB, TELB \*/

/\* Append the new codes in 23E \*/

**For n**=1 to j

**IF** Instruction[n]. Additional Information **NOT IsEmpty** THEN

**IF** **Length**(Instruction[n]. Additional Information) > 30 THEN

Instruction[n]. Additional Information = **Concatenate**(**Substring**(Instruction[n]. Additional Information,1, 29), “+”)

**ENDIF**

23E[n]=**Concatenate**(Instruction[n].Instruction Code

“/”, Instruction[n]. Additional Information)

**ELSE**

23E[n] = Instruction[n].Instruction Code

**ENDIF**

**n=n+1**

/\* Check if a code is present several times. As in MT, a code with same value cannot be present multiple times, keep the 23E occurrence with additional information and delete in the target message the others with the same codeword. If both have additional information, keep the first one and do not translate the other same codes but use the flag Missing Information to inform that all codes were not translated from MX \*/

Example, after translation of the business information, the following result might occur:

:23E:PHOB

:23E:PHOB/Call Number 1234567

:23E:PHOB/Call Number 4567890

The presence of the 3 lines in MT will generate an error because several times the same code word is not allowed.

So only the second line should be present in the target message with the Flag\_MissingInformation = “True”

### 4.3.13 MX\_To\_MTCharSet

**Name**

MX\_To\_MTCharSet

**Business description**

The function scans an MX text string and replaces all the extra characters not supported in FIN by Fullstop

**FIN character set is the following :**

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

/ - ? : ( ) . , ' +

CBPR+ allows extended character set in some elements like parties Name and addresses, Remittance Information, … These characters must be removed and replaced by a Fullstop

Additional special characters are supported:

!#$%&\*=^\_`{|}~";<>@[\]

The following escape sequence of char must also be identified in the data and replaced by a Fullstop.

&amp;

&quot;

&lt;

&gt;

**Format**

**MX\_To\_MTCharSet**(MXText; MTtext)

**Input**

MXText: MX string

**Output**

MTText : MT string of FIN char only

**Preconditions** None.

**Formal description**

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

/ - ? : ( ) . , ' +

ReplacingChar : string

ReplacingChar = “.”

**For n** = 1, **Length**(MXString)

**IF** **Substring**(MXText,n,1) **NOT IsInList**

{ a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

/ - ? : ( ) . , ' + }

/\* where “{“ and “}” are delimiters for the list and not part of the allowed characters. \*/

THEN

MTText = **Concatenate**(**Substring**(MXText,1, n-1), ReplacingChar,

**Substring**(MXText,n+1))

**n= n +1**

/\* Search for escape sequence of characters and replace by ReplacingChar \*/

For each pattern (MXPattern)InList

&amp;

&quot;

&lt;

&gt;

**Repeat**

**IF IsPresentPattern**(MXText, MXPattern) THEN

MXText = **ReplacePattern**(MXText, MXPattern, ReplacingChar)

/\* Character replacement must be reported in Error handling \*/

**ENDIF**

### 4.3.14 MX\_To\_MT72FullField2

**Name**

MX\_To\_MT72 FullField2

*/\* Note for developers. This function is similar to MX\_To\_MT72FullField but still some differences. For example, Code for IntermediaryAgent2,3 and PreviousInstructingAgent1,2,3 is the same. InstructionForCreditorAgent => differences \*/*

**Business Description**

The function combines elements from the MX message that must be carried to the next agent in order to allow the processing of the payment all along the chain.

Depending on the room left and the presence of the elements in the physical message, the following priority will be applied with possible missing or truncated information:

Priority 1 : IntermediaryAgent2,3 (/INTA/)\*

Priority 2 : PaymentTypeInformation/ServiceLevel (/SVCLVL/)\*

Priority 3 : PaymentTypeInformation/LocalInstrument (/LOCINS/)\*

Priority 4 : PaymentTypeInformation/CategoryPurpose (/CATPURP/)\*

Priority 5 : InstructionForCreditorAgent ([/UDLC/], /ACC/, /PHONBEN/, /TELEBEN/)

Priority 6 : Purpose (/PURP/)\*

Priority 7: InstructionForNextAgent (excluded “/FIN53/BIC” structure ~~and /FIN54/~~)

Priority 8 : DebtorAgent (/INS/)

Priority 9 : PreviousInstructingAgent1,2,3

Priority10 : RemittanceInformation (included code /BNF/ and /TSU/ from originating MT, if originating MX then info will be translated with /BNF/ code)

\*means new code words to be used in Field72

/UDLC/ only if pacs.009 CORE is used with pacs.009 ADV.

If a data is truncated (ie., partially present), “+” is added at the end of the string;

If an information cannot be copied in field 72 (eg., PreviousIntructingAgent2,3), an indicator Flag\_MissingInformation is returned with value “True”.

The function can be used to convert for the first time an MX message created in MX to an MT message or to convert a MX message already converted previously in MT in a sequence like MX to MT to MX or to convert a MX message initiated in MT.

**IntermediaryAgent2,3** can only be present if the original message is a MX message. That information must be carried to continue the payment process and a new code will be used /INTA/ in order to fit with field 72 structure. This code can be repeated max 2 times. It can be understood as an instruction for IntermediaryAgent1 (/INT/)on how to continue the chain but in order to indicate it is a new information coming from an original MX message, a new code word is used.

IntermediaryAgent2,3 must be identified at a minimum with a BIC OR (Name [and TownName and Country])for cross border transactions or by ClearingSystemMemberIdentification for domestic transactions.

BIC is the preferred option for the translation.

IF Country is present, Name is truncated after 34 characters if length > 35 else Name is truncated after 69 characters if length > 70 (and added “+” for truncation indication). Preference is given to BICFI.

The ISO Clearing System Identification is not converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

Proposal MT72 field template for IntermediaryAgent2,3:

/INTA/BIC OR /INTA/Name [**(**Country**(**TownName] OR /INTA/ClearingSystemMemberIdentification

Where “(“ is used as a separator easy to identify in a string by searching a pattern “(“ followed by exactly 2 alphabetic characters which must be an ISO country code followed by “(“.

Country and TownName will be provided if present in the MX

If IntermediaryAgent2,3 are present, with the above information, 6 lines might be consumed and no room left for other information as shown below, if present in MX. So the following truncation is done:

IF Name is translated, Name length is calculated in order to fill max 2 lines in MT (including the MX code word)

ServiceLevel (ISO code or Proprietary) is translated to Field 72 with code word /SVCLVL/. For the ISO codes, only codes different from “G00n” where n is an integer, are translated to field 72. gpi Codes (G00n) are translated to FIN Block3/EndToEndReference/ServiceTypeIdentifier in a different function. Up to 3 occurrences /SVCLVL/ are possible, although it is more likely that maximum 2 will be translated to field 72.

LocalInstrument (ISO code or Proprietary) is translated to Field 72 with code word /LOCINS/. Only one occurrence is allowed.

CategoryPurpose (ISO code or Proprietary) is translated to Field 72 with code word /CATPURP/. Only one occurrence is allowed.

**InstructionForCreditorAgent**, when InstructionInformation is used with a code, “PHOB” or “TELB”, InstructionInformation is related to the code “PHOB” or “TELB”.

When InstructionInformation is used with no code in one or both occurrences of InstructionForCreditorAgent, the information is translated to /ACC/. Up to 2 occurrences of InstructionForCreditorAgent is allowed.

Information from InstructionForCreditorAgent is translated following the priority : /ACC/, /PHONBEN/, /TELEBEN/

Purpose (ISO code or Proprietary) is translated to Field 72 with code word /PURP/. Only one occurrence is allowed.

**InstructionForNextAgent**.**InstructionInformation** may be present without code in case the information is originated from an MX message or could contain codes like /REC/, /INT/ or /ProprietaryCode/ if the element results from a previous translation MT to MX or if the user replicates in MX the MT practices from field 72. When InstructionInformation is originated from MX with no code, it will be translated with the code /REC/ to MT.

**The codes /FIN53/** followed by BIC can be found in InstructionForNextAgent.InstructionInformation from a previous translation MT to MX. Although they should have been be removed from the receiver side when the next leg of the payment was created unless the same code is used in MX to mimic the same serial scenario (refer to MT202 Settlement Method&Agent sheet) From MX to MT there is no check on the BIC’s value following the code. But this is not encouraged as it is a misuse of the standards. Translation caters for the handling of that code but outside of this current function because the prioritization of the information to be translated to field 72 does not apply to code /FIN53/. A dedicated function has been created MX\_To\_MT53A. Refer also to SubfunctionInstructionForNextAgent for more information on the extraction of /FIN53/ related information.

**PreviousInstructingAgent1,2,3** exist already in MT and are transported with the code /INS/followed by BIC or Name in field 72 or they could be defined for the first time in an original MX message. From an original MX message, if the postal address is present, it is recommended to use a structured postal address while if the presence of these agents in MX is already resulting from an MT to MX translation, only a unstructured postal address (eg., AddressLine) is present with value “NOTPROVIDED”. Nevertheless the translation MX to MT will ignore the PostalAddress as this information is currently not available in MT. BIC is the preferred option for the translation.

In case both BIC and Name are absent in MX, the ClearingSystemMemberIdentification will be translated in order to cater for domestic transactions which might only use the ClearingSystemMemberIdentification. This will only happen if the original message is MX.

The ISO Clearing System Identification is not converted into the MT Clearing System Identification (eg.,”AUBSB” is not converted to “AU”).

Name and ClearingSystemMemberIdentification are truncated if length is greater than 210 characters. A sign “+” is added to indicate truncation.

DebtorAgent is translated in the same way as PreviousInstructingAgent with code /INS/ but with higher priority than PreviousInstructingAgent. DebtorAgent will only be present if the payment is originated in MX in which case the DebtorAgent should not be filled also in PreviousInstructingAgent in order to avoid duplicate in translation.

RemittanceInformation is translated to Field 72. Only one occurrence is allowed in CBPR+ with max 140 characters. Code /BNF/ and /TSU/ can be present coming from previous MT to MX translation. Each code is expected to be present maximum once. If information is present (at the beginning on the RI string) with no code, it will be concatenated with information following code /BNF/ or if the code is not yet used, then it will be created to translate to field 72.

**Format**

MX\_To\_MT72 FullField2

**Input**

IntermediaryAgent2 , IntermediaryAgent3, PreviousInstructingAgent1, PreviousInstructingAgent2, PreviousInstructingAgent3, DebtorAgent typed BranchAndFinancialInstitutionIdentification6

InstructionForCreditorAgent typed InstructionForCreditorAgent2

InstructionForNextAgent typed InstructionForNextAgent1

CategoryPurpose typedCategoryPurpose1Choice

Purpose typed Purpose2Choice

RemittanceInformation typed RemittanceInformation2

CreditTransferTransactionInformation/PaymentTypeInformation typed PaymentTypeInformation28

MXSettlementMethod typed SettlementMethod1Code

BusinessApplicationHeader

**Output**

MT72: content of an MT field 72 in lines of maximum 6 \* 35 characters.

**Preconditions**

At least one of the input parameters is present (eg., not empty)

**Formal description**

/\* Local variables :

MXAgent typed BranchAndFinancialInstitutionIdentification6, MTAgent is a string MTTemp72 has a structure 3\*35 char, PresentInfo is boolean, MXInstruction is a string, MTInstruction is a string, NumberofEmptyLines is a integer , MTInstructionLength is integer, NumberOf char is integer, MTNoCodeInstruction is a string, MTInstructionForNextAgent is a string\*/

**/\* Manage the priorities and depending on the room left additional information is translated to Field 72\*/**

**/\* Function call section \*/**

**IF** (intermediaryAgent2) **IsPresent** THEN

Call **SubfunctionIntermediaryAgents**

**ENDIF**

**IF** PaymentTypeInformation/ServiceLevel **IsPresent** THEN

**Call** **SubfunctionServiceLevel**

**ENDIF**

**IF** PaymentTypeInformation/LocalInstrument IsPresent THEN

**Call** **SubfunctionLocalInstrument**

**ENDIF**

**IF** PaymentTypeInformation/CategoryPurpose **IsPresent** THEN

**Call** **SubfunctionCategoryPurpose**

**ENDIF**

**IF** InstructionForCreditorAgent **IsPresent** THEN

**Call** **SubfunctionInstructionForCreditorAgent**

**ENDIF**

**IF** Purpose **IsPresent** THEN

**Call** **SubfunctionPurpose**

**ENDIF**

**IF** InstructionForNextAgent **IsPresent** THEN

**Call** **SubfunctionInstructionForNextAgent**

**ENDIF**

**IF** DebtorAgent **IsPresent** THEN

**Call** **SubfunctionDebtorAgent**

**ENDIF**

**IF** PreviousInstructingAgent **IsPresent** THEN

**Call** **SubfunctionPreviousInstructingAgent**

**ENDIF**

**IF** RemittanceInformation **IsPresent** THEN

**Call SubfunctionRemittanceInformation**

**ENDIF**

**/\* EndFunction call section \*/**

**/\* Subfunctions specifications \*/**

**SubfunctionIntermediaryAgents**

**/\* Extract IntermediaryAgents \*/**

**IF** IntermediaryAgent2 **IsPresent** THEN

MXAgent = MXIntermediaryAgent2

/\* Extract information, from MXAgent \*/

**MX\_To\_MTAgent**(MXAgent,Length(“/INTA/”); MTAgent)

/\* Format the information for field MT72 \*/

**IF** **Length**(MTAgent) > 0 THEN

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ENDIF**

MTAgent = **Concatenate** (“/INTA/”, MTAgent)

NumberOfEmptyLines = 6 – ReturnFirstLineEmpty (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTAgent, MT72;MT72)

**ENDIF**  /\* LENGTH(MTAgent) > 0 \*/

**ENDIF** /\* IntermediaryAgent2 IsPresent \*/

**IF** IntermediaryAgent3 IsPresent THEN

MXAgent = MXIntermediaryAgent3

/\* Extract information, from MXAgent \*/

**MX\_To\_MTAgent**(MXAgent, Length(“/INTA/”; MTAgent)

/\* Format the information for field MT72 \*/

**IF** **Length**(MTAgent) > 0 THEN

/ \*check room left \*/

**IF** ReturnFirstLineEmpty (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ENDIF**

MTAgent = **Concatenate** (“/INTA/”, MTAgent)

NumberOfEmptyLines = 6 – ReturnFirstLineEmpty (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTAgent, MT72;MT72)

**ENDIF**  /\* Length(MTAgent) > 0 \*/

**ENDIF** /\* IntermediaryAgent3 IsPresent \*/

/\*with the IntermediaryAgent2,3, min 2 lines and max 6 lines can be consumed. \*/

**SubfunctionServiceLevel**

/\* LocalVariables

MXServiceLevel :string \*/

**For n**= 1 to 3

**IF** CreditTransferTransactionInformation/PaymentTypeInformation/ServiceLevel[n] **IsPresent** THEN

**IF** ServiceLevel[n].Proprietary **IsPresent** THEN

MXServiceLevel = ServiceLevel[n].Proprietary

**ELSE**

**IF** ServiceLevel[n].Code **IsPresent** AND ServiceLevel[n].Code <> **Pattern**(G00n) THEN

/\* n being an integer. Example of gpi service identifier value “G001” \*/

MXServiceLevel = ServiceLevel[n].Code

**ENDIF**

**ENDIF**

**IF** **Length**(MXServiceLevel) > 0 THEN

/ \*check room left \*/

**IF** ReturnFirstLineEmpty (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ENDIF**

MXServiceLevel = **Concatenate** (“/SVCLVL/”, MXServiceLevel)

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MXServiceLevel, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** Length(MXServiceLevel) > 0 \*/

**ENDIF**  /\* ServiceLevel[n] isPresent \*/

**Next** n

**SubfunctionLocalInstrument**

/\* LocalVariables

MXLocalInstrument : string \*/

**IF** CreditTransferTransactionInformation/PaymentTypeInformation/LocalInstrument **IsPresent** THEN

**IF** LocalInstrument.Proprietary **IsPresent** THEN

MXLocalInstrument = LocalInstrument.Proprietary

**ELSE**

MXLocalInstrument = LocalInstrument.Code

**ENDIF**

**IF** **Length**(MXLocalInstrument) > 0 THEN

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ENDIF**

MXLocalInstrument = **Concatenate** (“/LOCINS/”, MXLocalInstrument)

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MXLocalInstrument, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** Length(MXLocalInstrument) > 0 \*/

**ENDIF**  /\* LocalInstrument isPresent \*/

**SubfunctionCategoryPurpose**

/\* LocalVariables

MXCategoryPurpose : string \*/

**IF** CreditTransferTransactionInformation/PaymentTypeInformation/CategoryPurpose **IsPresent** THEN

**IF** CategoryPurpose.Proprietary **IsPresent** THEN

MXCategoryPurpose = CategoryPurpose.Proprietary

**ELSE**

MXCategoryPurpose = CategoryPurpose.Code

**ENDIF**

**IF** **Length**(MXCategoryPurpose) > 0 THEN

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ENDIF**

MXCategoryPurpose = **Concatenate** (“/CATPURP/”, MXCategoryPurpose)

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MXCategoryPurpose, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** LENGTH(MXCategoryPurpose) > 0 \*/

**ENDIF** /\* CategoryPurpose isPresent \*/

**SubfunctionPurpose**

/\* LocalVariables

MXPurpose : string \*/

**IF** Purpose **IsPresent** THEN

**IF** Purpose.Proprietary **IsPresent** THEN

MXPurpose = Purpose.Proprietary

**ELSE**

MXPurpose = Purpose.Code

**ENDIF**

**IF** **Length**(MXPurpose) > 0 THEN

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ENDIF**

MXPurpose = **Concatenate** (“/PURP/”, MXPurpose)

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MXPurpose, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** Length(MXPurpose) > 0 \*/

**ENDIF** /\* Purpose isPresent \*/

**SubfunctionInstructionForNextAgent**

/\* Local variable

RECIndicator : Boolean

TempREC : string \*/

RECIndicator = “false”

TempREC = “/TempREC/”

/\* Build a string to concatenate the 6 occurrences from InstructionForNextAgent.InstructionInformation. If the payment is originated in MX, there is no guideline on how to fill InstructionInformation. It may be a mix of information with codewords and without codeword. It may be that information without codeword uses a new occurrence. In the extraction of the information, the assumption is made that if an occurrence is complete (max 35 char) the next occurrence is the continuation unless the next line starts with a codeword. So,in the concatenation of the 6 occurrences, if an occurrence is not complete, a tempory codeword /TempREC/ will be added before the next occurrence if the next occurrence does not start with a codeword, to delimitate the information.

If an occurrence contains a codeword and related information is followed by information not related to the codeword,there is no way in translation to identify it is not part of the same codeword. For example /BNF/BeneficiaryName Text1. Text1 will be considered as part of /BNF/

A special treatment is needed for codeword /FIN53/ generated from MT to MX translation. So first, it is needed to extract /FIN53/ and following information if present.

If information is a valid BIC (and only a BIC, not a BIC and textual information) and if the SettlementMethod is INGA or INDA,there are 2 scenarios in which cases the BIC is translated to 53A/Identifier code (refer to excel MX to MT and also MT to MX translation for more information):

-The SettlementAccount is present OR

-The SettlementAccount is absent AND the first 6 char of the BIC equals the first 6 char of the Sender or the first 6 char of the Receiver

If the conditions of one of the above scenarios are met, the BIC is translated to 53A by MX\_To\_MT53A function. And therefore /FIN53/BIC can be deleted from InstructionForNextAgent.

In all the other cases, the codeword /FIN53/ and related information is translated to Field 72 as any other code words. In particular, a case like “/FIN53/ValidBICText” or “/FIN53/ValidBIC Text” is treated as any other generic codes and the ValidBIC is not treated as a BIC but is part of the Text information and all is translated to field 72 to avoid data lost. IF the SettlementMethod is “COVER” (/FIN53/ is then not expected but for smooth translation it is handled in this function),/FIN53/ will be treated as any other generic codes and translated to Field 72 to avoid data lost.

**IMPORTANT** - As the same function is used for pacs.008, pacs.009 CORE, pacs.009 COVE and pacs.009 ADV, the SettlementMethod must be set to “COVE” when the translation is done from **pacs.009 COVE/UnderlyingCustomerCreditTransfer/InstructionForNextAgent**. This is done in the excel spreadsheet. The meaning of **forcing** the SettlementMethod parameter to “COVE” is to force the translation of /FIN53/Information to Field 72 and avoid translation to Field 53A which is absent in Seq B. note that any other valid value different from INGA or INDA could have been used as well (ie, CLRG)

By default for the other source messages, the SettlementMethod is passed as such as a parameter.

The remaining string will be analysed to extract the other codes to be translated to Field 72\*/

**IF** InstructionForNextAgent[1] **IsPresent** THEN

MTInstructionForNextAgent = InstructionForNextAgent[1].InstructionInformation

**For m** = 2 to 6

**IF** InstructionForNextAgent[m] **IsPresent** THEN

**IF**(**Length**(InstructionForNextAgent[m-1])< 35 AND InstructionForNextAgent[m].InstructionInformation **DOES NOT** **START with** “/GenericMax8c/”) THEN

MTInstructionForNextAgent = **Concatenate** (MTInstructionForNextAgent, **TempREC**, InstructionForNextAgent[m].InstructionInformation)

**ELSE**

MTInstructionForNextAgent = **Concatenate** (MTInstructionForNextAgent, InstructionForNextAgent[m].InstructionInformation)

**ENDIF**

**ENDIF**

**Next** m

/\* Extract /FIN53/ and related information if present. Other possible codes are /INT/, /REC/, (also from MT202 : /PHON/, /PHONIBK/, /TELE/, /TELEIBK/), or /ProprietaryCode/ meaning any other proprietary codes defined with maximum 8 alpha numeric upper case characters. Let’s name the codes /GenericMax8c/ \*/

**IF** **IsPresentPattern**(MTInstructionForNextAgent, "/FIN53/") THEN

MTInstructionTemp = **ExtractBetweenPattern**(MTInstructionForNextAgent,”/FIN53/”, {“/GenericMax8c/”, TempRec})

**IF IsValidBIC**(MTInstructionTemp) AND MXSettlementMethod = “INDA” OR “INGA” THEN

**IF IsPresent**(SettlementAccount) **OR**

**{IsAbsent**(SettlementAccount) **AND (Substring**(MTInstructionTemp,1,6) = **Substring**(BAH/From/financialInstitutionIdentification/BICFI,1,6) **OR** **Substring**(MTInstructionTemp,1,6) = **Substring**(BAH/To/financialInstitutionIdentification/BICFI,1,6))**}**

/\* BIC is translated with MX\_To\_MT53A and then removed from the string to analysed. Note that it is not expected to have code /FIN53/ present several times. In the assumption this happens only the first occurrence is checked to extract a valid BIC. If the second occurrence of /FIN53/ carries a valid BIC, it is translated to Field 72 meaning the information is not lost. But there is no rationale to have /FIN53/ present several times \*/

MTInstructionTemp = **Concatenate**(“/FIN53/”, MTInstructionTemp)

ReplacePattern(MTInstructionForNextAgent, MTInstructionTemp,“”)

**ENDIF ENDIF**

**ENDIF** /\* ENDIF IsPresentPattern(MTInstruction, "/FIN53/") \*/

**IF** MTInstructionForNextAgent **NOT IsEmpty** THEN

**{**

/\* Other codes remaining in MTInstructionForNextAgent will be translated to Field 72 \*/

/\*Search for patterns like **/REC/, /INT/, /FIN53/ and in general /max 8char/** in the MTInstruction string.

If found, extract the pattern and the following related information up to the next pattern “/GenericMax8c/” or TempREC, if any, or up to the end in case no next pattern “/GenericMax8c/” or no TempREC is found. Store the pattern and related information in a table with index **i**  (eg., MTInstruction[1], MTInstruction[2] where MTInstruction**[i]** contains info in the format “/Code/InstructionInformation” ). Delete each pattern found and related information before starting the next search.

At this stage, the remaining string MTInstructionForNextAgent must not contain any code word like **/REC/, /INT/, /FIN53/ or any other /max 8char/**. It should have a structure like [/TempREC/]Text1/TempREC/Text2/TempREC/Text3…

Copy that information to **MTNoCodeInstruction after having replaced “/TempREC/” by SPACE and TrimLeft(MTNoCodeInstruction,SPACE)** \*/

/\* Either there is a table or MTNoCodeInstruction is not empty or both\*/

/\* i is the last index in the MTInstruction table\*/

**IF** MTNoCodeInstruction **NOT IsEmpty** THEN

/\* MTNoCodeInstruction is concatenated with element with code /REC/ in the table . Either the table element exists or it must be created \*/

**IF** i > 0 THEN /\* Table exists \*/

**For n** = 1 to i

**IF** **Substring**(MTInstruction[n],1, 5) = “/REC/” THEN

{

MTInstruction[n] = **Concatenate** (MTInstruction[n],” “ , MTNoCodeInstruction)

RECIndicator = “True”

Exit loop }

**ELSE**

/\* Do nothing \*/

**ENDIF**

**n** = n+ 1

**ENDIF** /\* IF i > 0 \*/

**IF** RECIndicator = “False” THEN

/\* either no table or no /REC/ in the table \*/

i=i+1 /\* add new element \*/

MTInstruction[i] = **Concatenate** (“/REC/”, MTNoCodeInstruction)

**ENDIF** /\* RECIndicator = “False” \*/

/\* At least one element in the table \*/

/\* The information stored in table must be formatted following MT72 structure \*/

**ENDIF** /\*MTNoCodeInstruction NOT IsEmpty \*/

**For** n = 1 to i

**{**

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ELSE**

NumberOfEmptyLines = 6 - **ReturnFirstLineEmpty** (MT72, 6)+1

**AppendComplexMT72**(NumberOfEmptyLines, MTInstruction[n] ,MT72; MT72)

**ENDIF**

**}**

**n = n +1**

**}** /\* IF MTInstructionForNextAgent NOT IsEmpty \*/

**ENDIF**

**SubfunctionInstructionForCreditorAgent**

/\* Extract the codes and related information to a Table[i] with component Code and InstructionInformation. Codes are /PHOB/, /TELB/ and information with no Code will be associated to /ACC/ in MT \*/

/\* The function also searches for codes /PHOB/ and /TELB/ in InstructionInformation as a side effect of a previous MT to MX translation (refer to MT\_To\_MXInstructionForCreditorAgent). Although it is unlikely to use in the same time /PHOB/, /TELB/ (ie., is even forbidden in field 23E in MT103)\*/

/\* In pacs.009 **CORE** **used to settle a previously sent pacs.009 ADV**, a special code has been introduced to carry information about the Underlying Creditor . The expected format is /UDLC/BIC or /UDLC/Name and Postal Address. The codeword /UDLC/ and information are translated as such to Field 72 with priority 1 versus codes /ACC/, /PHONBEN/ and /TELEBEN/ (code values in MT).

It is expected that only 1 occurrence of InstructionInformation can get /UDLC/ and the code word must start the occurrence of InstructionInformation. It is also expected that the latest available occurrence will be used in order to avoid cases where /UDLC/ information fills in the first occurrence with exact 140 char and the second occurrence is wrongly assimilate as the continuity of the first occurrence. Nevertheless, these usage rules are not validated. So in order to build a strong translation, the function will look for other possible configurations but with the following principles:

If in an occurrence of InstructionForCreditorAgent, InstructionForCreditorAgent.Code is present, then the InstructionInformation is expected to be linked to the code and will be analysed alone.

If there are 2 occurrences of InstructionForCreditorAgent and none of them has the MX code element, then the 2 InstructionInformations are concatenated to optimise the extraction of the information containing possibly the other codes /UDLC/, /PHOB/ and /TELB/.

If the first occurrence is less than 140 char and the second occurrence does not start with /UDLC/, /PHOB/ and /TELB/, the 2 occurrences are concatenated with /TempACC/ in between to indicate that the second occurrence is not the continuation of the first one.

The information from /TempACC/ will then be translated to field 72 with code /ACC/after possible concatenation with other information having the same “semantic” /ACC/.

If the code /UDLC/ occurs more than one time (max twice), information is concatenated with a space with the information extracted from the other /UDLC/ code. /UDLC/ is assumed to be the only code that can be repeated in InstructionInformation.

The information not related to one of the code is translated with the code /ACC/ to Field 72 and possibly concatenated if several piece of such information occur.

The information is translated to field 72 with the following priority and depending on the room left:

1. /UDLC/
2. /ACC/
3. /PHONBEN/
4. /TELEBEN/

See example 8 and 9 for possible side effect in translation.

Example1

InstructionForCreditorAgent[1].Code Is absent

InstructionForCreditorAgent[1].InstructionInformation = Text1/UDLC/Name and Address

InstructionForCreditorAgent[2].Code Is absent

InstructionForCreditorAgent[2].InstructionInformation = Text2

After extraction:

/UDLC/Name and Address

/ACC/Text1 Text2

Example2

InstructionForCreditorAgent[1].Code = PHOB

InstructionForCreditorAgent[1].InstructionInformation = Text1/UDLC/Name and Address

InstructionForCreditorAgent[2].Code Is absent

InstructionForCreditorAgent[2].InstructionInformation = Text2

After extraction:

/UDLC/Name and Address

/ACC/Text2

/PHONBEN/Text1

Example 3

InstructionForCreditorAgent[1].Code Is absent

InstructionForCreditorAgent[1].InstructionInformation = Text1/UDLC/Name and Address

InstructionForCreditorAgent[2].Code Is absent

InstructionForCreditorAgent[2].InstructionInformation = Text2/UDLC/BIC

After extraction:

/UDLC/Name and Address BIC

/ACC/Text1 Text2

Example 4

InstructionForCreditorAgent[1].Code = PHOB

InstructionForCreditorAgent[1].InstructionInformation is absent

InstructionForCreditorAgent[2].Code Is absent

InstructionForCreditorAgent[2].InstructionInformation = Text2/UDLC/BIC

After extraction:

/UDLC/BIC

/ACC/Text2

/PHONBEN/

Example 5

InstructionForCreditorAgent[1].Code = PHOB

InstructionForCreditorAgent[1].InstructionInformation = /UDLC/BIC

After extraction:

/UDLC/BIC

/PHONBEN/

Example 6

InstructionForCreditorAgent[1].Code = PHOB

InstructionForCreditorAgent[1].InstructionInformation is absent

InstructionForCreditorAgent[2].Code = TELB

InstructionForCreditorAgent[2].InstructionInformation is absent

After extraction:

/PHONBEN/

/TELEBEN/

Example 7

InstructionForCreditorAgent[1].Code is absent

InstructionForCreditorAgent[1].InstructionInformation uses 140 char and the last 3 characters are as following : TEXT1… /PH

InstructionForCreditorAgent[2].Code is absent

InstructionForCreditorAgent[2].InstructionInformation = OB/Text2

After extraction:

/ACC/Text1….

/PHONBEN/Text2

Example8

InstructionForCreditorAgent[1].Code Is absent

InstructionForCreditorAgent[1].InstructionInformation = /UDLC/Name and Address => uses exact 140 char

InstructionForCreditorAgent[2].Code Is absent

InstructionForCreditorAgent[2].InstructionInformation = Text2

After extraction:

/UDLC/Name and AddresText2

**In this example, the second occurrence of InstructionInformation is handled as the continuation of the first occurrence. So if by accident the first occurrence is full, and the second occurrence has nothing to do with the code /UDLC/, to avoid such a case, it is recommended to start the first occurrence with Text2 and then start the second occurrence with /UDLC/. Then the translation result will be correct as shown in Example 9**

Example9

InstructionForCreditorAgent[1].Code Is absent

InstructionForCreditorAgent[1].InstructionInformation = Text2

InstructionForCreditorAgent[2].Code Is absent

InstructionForCreditorAgent[2].InstructionInformation = /UDLC/Name and Address => uses exact 140 char

After extraction:

/UDLC/Name and Address

/ACC/ = Text2

/\* Variables \*/

CodeTable[] : table of codes

CodeList : list of codes {/UDLC/, /PHOB/, /TELB/}

ExtractedCodeTable[], ACCTable[1], UDLCTable[1], PHOBTable[1],TELBTAble[1] : table made of 2 components (Code and Text) components

m,i,j,k, t, u : integer

CodeTable[1] = “/UDLC/”

CodeTable[2] = “/PHOB/”

CodeTable[3] = “/TELB/”

CodeTable[4] = “/TempAcc/”

CodeTable[5] = “/UDLC/”

/\* CodeTable[5] allows to cater for the case where “/UDLC/” is used twice in InstructionInformation \*/

**IF IsPresent**(InstructionForCreditorAgent[1] AND **IsAbsent**(InstructionForCreditorAgent[1].Code) AND

**IsPresent**(InstructionForCreditorAgent[2] AND **IsAbsent**(InstructionForCreditorAgent[2].Code) **THEN**

/\* 2 occurrences of InstructionInformation are concatenated \*/

**{IF Length**(InstructionForCreditorAgent[1].InstructionInformation)<140 **AND NOT WithinList**(**Substring**(InstructionForCreditorAgent[2].InstructionInformation),1,6), CodeList)THEN

MXInstruction = **Concatenate**(InstructionForCreditorAgent[1].InstructionInformation,”/TempACC/”, InstructionForCreditorAgent[2].InstructionInformation)

**ELSE**

MXInstruction = **Concatenate**(InstructionForCreditorAgent[1].InstructionInformation, InstructionForCreditorAgent[2].InstructionInformation)

**ENDIF}**

InstructionCode = “”

**call SubfunctionExtractCode**(InstructionCode,CodeTable[], MXInstruction; ExtractedCodeTable[])

**ElSE**

/\* Each occurrence of InstructionInformation is analysed separately in thesubfunction. Subfunction described below \*/

**Call SubfunctionAnalysePer1Occurrence2**

**ENDIF**

/\* Build the strings out of the ExtractedCodeTable for each code /UDLC/, /ACC/, /PHONEBEN/ and /TELEBEN/ by concatenating information related to the same code with a space between the pieces of information \*/

/\* There is one table ExtractedCodeTable[] which might contain several times the same code. The related information linked to a same code is concatenated with a space in order to copy the code only once to Field 72 \*/

For m = 1 to **NumberOfOccurrences**(ExtractedCodeTable[])

/\* Fill in [], ACCTable[1], UDLCTable[1], PHOBTable[1],TELBTAble[1] \*/

**IF** ExtractCodeTable[m].Code = “/TempACC/”OR **IsEmpty**(ExtractCodeTable[m].Code)THEN

**{**

ACCTable[1].Code = “/ACC/”

**SubfunctionSameCodeInfoConcatenation** (ExtractCodeTable[m].Text, ACCTable[1].Text; ACCTable[1].Text)

**}**

**ELSEIF** ExtractCodeTable[m].Code = “/UDLC/”

**{**

UDLCTable[1].Code = “/UDLC/”

**SubfunctionSameCodeInfoConcatenation** (ExtractCodeTable[m].Text, UDLCTable[1].Text; UDLCTable[1].Text)

**}**

**ELSEIF** ExtractCodeTable[m].Code = “/PHOB/”

**{**

PHOBTable[1].Code = “/PHONBEN/”

**SubfunctionSameCodeInfoConcatenation** (ExtractCodeTable[m].Text, PHOBTable[1].Text; PHOBTable[1].Text)

**}**

**ELSEIF** ExtractCodeTable[m].Code = “/TELB/”

**{**

TELBTable[1].Code = “/TELEBEN/”

**SubfunctionSameCodeInfoConcatenation** (ExtractCodeTable[m].Text, TELBTable[1].Text; TELBTable[1].Text)

**}**

**ENDIF** /\* no other values expected \*/

**Next** m

/\* Sort Table to prioritize the information as following /UDLC/, /ACC/, /PHONBEN/, /TELEBEN/

/\* There is at least one element in the Table \*/

i = **NumberOfOccurrences**(UDLCTable[])

j = **NumberOfOccurrences**(ACCTable[])

k = **NumberOfOccurrences**(PHOBTable[])

m = **NumberOfOccurrences**(TELBTable[])

t = 0

**IF** i > 0 Then

**For** u = 1 to i

Table[u].Code = UDLCTable[u].Code

Table[u].InstructionInformation = UDLCTable[u].Text

**Next** u

**ENDIF**

t = i

**IF** j > 0 Then

**For** u = 1 to j

t = t+1

Table[t].Code = ACCTable[u].Code

Table[t].InstructionInformation = ACCTable[u].Text

**Next** u

**ENDIF**

**IF** k > 0 Then

**For** u = 1 to k

t = t+1

Table[t].Code = PHOBTable[u].Code

Table[t].InstructionInformation = PHOBTable[u].Text

**Next** u

**ENDIF**

**IF** m > 0 Then

**For** u = 1 to m

t = t+1

Table[t].Code = TELBTable[u].Code

Table[t].InstructionInformation = TELBTable[u].Text

**Next** u

**ENDIF**

/\* Copy information to field 72. Code /PHONBEN/ and /TELEBEN/ can be alone without additional information. /ACC/ must be followed by information. /UDLC/ is expected to be followed by information but in case no information is linked to /UDLC/, the code is translated as such and alone. It is nevertheless a misuse of the code /UDLC/. \*/

**For j** = 1 to t

**IF** **ReturnFirstLineEmpty**(MT72,6) = 0 THEN

/\* no line left \*/

Flag\_MissingInformation = “true”

EXIT function MX\_To\_MT72FullField2

**ELSE**

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

Temp72 = **Concatenate**(Table[j].Code,Table[j].InstructionInformation)

**AppendComplexMT72**(NumberOfEmptyLines, Temp72, MT72;MT72)

**ENDIF**

**Next j**

**SubfunctionSameCodeInfoConcatenation** (ExtractedString, TableText;TableText)

**IF NOT IsEmpty**(ExtractedString)

**IF Length**(TableText)> 0 THEN

TableText = **Concatenate**(TableText, SPACE, ExtractedString)

**ELSE**

TableText = ExtractedString

**ENDIF**

**ENDIF**

**SubfunctionAnalysePer1Occurrence2**

/\* Each occurrence of InstructionForCreditorAgent is analysed separately. For each occurrence at least either MX Code is present or MX InstructionInformation is present. Both can be present \*/

/\* local Variables

j,k,m,t:integer

MXInstruction, InstructionCode : string

ExtractedCodeTable1[], ExtractedCodeTable2[] : table made of Table.Code and Table.Text

\*/

**For j** = 1 to NumberOfOccurences(InstructionForCreditorAgent)

**IF IsPresent**(InstructionForCreditorAgent[j].InstructionInformation THEN

**{** MXInstruction = InstructionForCreditorAgent[j].InstructionInformation

**IF IsPresent**(InstructionForCreditorAgent[j].Code) THEN

InstructionCode = InstructionForCreditorAgent[j].Code

**ELSE**

InstructionCode = “”

**ENDIF**

**Call SubfunctionExtractCode**(InstructionCode,CodeTable[], MXInstruction; ExtractedCodeTable[])

/\* TempACC is not used when 2 occurrences are analysed separately, so will not be found in the string to analyse \*/

**}**

**ELSE**

/\* only MX Code is present without MX InstructionInformation \*/

ExtractedCodeTable[1].Code = InstructionForCreditorAgent[j].Code

ExtractedCodeTable[1].Text = “”

**ENDIF**

/\* Copy the ExtractedCodeTable[] temporarily \*/

**IF** j = 1 Then

**For k** = 1 to **NumberOf Occurrences**(ExtractedCodeTable[])

ExtractedCodeTable**1**[k] = ExtractedCodeTable[k]

**Next** k

**ELSE**

**For** k = 1 to **NumberOf Occurrences**(ExtractedTable[])

ExtractedCodeTable**2**[k] = ExtractedCodeTable[k]

**Next k**

**ENDIF**

**ENDIF**

**Next j**

/\* Merge the 2 tables \*/

j = NumberOfOccurrences(ExtractedCodeTable1[])

k = NumberOfOccurrences(ExtractedCodeTable2[])

m = NumberOfOccurrences(ExtractedCodeTable[])

/\* Reset ExtractedCodeTable[] \*/

**IF** m > 0 THEN

**For** t = 1 to m

ExtractedCodeTable[t].Code = “”

ExtractedCodeTable[t].Text = “”

**Next t**

**ENDIF**

**IF** j > 0 Then

**For** t = 1 to j

ExtractedCodeTable[t] = ExtractedCodeTable**1**[t]

**Next t**

**ENDIF**

**IF** k > 0 THEN

**For** t = 1 to k

j = j+1

ExtractedCodeTable[j] = ExtractedCodeTable**2**[t]

**Next** t

**ENDIF**

/\* ExtractedCodeTable contains all the information to be translated to Field 72 without code or with code /TempACC/, /UDLC/, /PHOB/, /TELB/ \*/

/\* End **SubfunctionAnalysePer1Occurrence2** \*/

**SubfunctionExtractCode**(InstructionCode,CodeTable[], MXInstruction; Table[])

/\* local variables

k,j,t:integer

MXText,Temp,RemainingString, FoundPatternText : string

Table[] is made of Table.Code and Table.Text

\*/

j = 0

RemainingString = MXInstruction

**For k** = 1 to 5

/\* k counts the number of code permutations\*/

**IF Length**(RemainingString) > 0 THEN

**IF IsPresentPattern**(RemainingString, CodeTable[1]) THEN

MXText = **ExtractBetweenPattern**(RemainingString, CodeTable[1],{CodeTable[2], CodeTable[3], CodeTable[4], CodeTable[5])

j= j + 1

Table[j].Code = CodeTable[1]

Table[j].Text = MXText

FoundPatternText = **Concatenate**(CodeTable[1], MXText)

RemainingString = **DeletePattern**(RemainingString, FoundPatternText)

**ENDIF**

/\* Permutation of the codes \*/

Temp = Code[1]

For t = 2 to 5

CodeTable[t-1] = CodeTable[t]

Next t

Code[5] = Temp

**ENDIF**

**Next** k

**IF Length**(RemainingString)> 0 **OR Length**(InstructionCode) > 0 THEN

j = j+1

Table[j].Code = InstructionCode

Table[j].Text = RemainingString

**ENDIF**

**/\* End SubfunctionExtractACC2 \*/**

**SubfunctionRemittanceInformation**

/\*Local variables

Table is made of Table.Code and Table.Information : string

TempInformation, MXRemittanceInformation : string

i : integer

BNFIndicator : boolean \*/

/\* Precondition is thatRemittanceInformation IsPresent (ie., Not empty) \*/

i = 0

BNFIndicator = “false”

MXRemittanceInformation = RemittanceInformation.Unstructured[1]

/\*Search for the /BNF/ codes. It is not expected there will be several ones. \*/

**IF** **IsPresentPattern**(MXRemittanceInformation, "/BNF/") THEN

TempInformation = **ExtractBetweenPattern**(MXRemittanceInformation,”/BNF/”, “/TSU/”)

**IF** TempInformation **NOT IsEmpty** THEN

/\* save information in the table \*/

i = i + 1

Table[i].Code = “/BNF/”

Table[i].Information = TempInformation

**ENDIF**

/\* Delete the code and string extracted \*/

MXRemittanceInformation = **DeletePattern**(MXRemittanceInformation,**Concatenate**(”/BNF/”, TempInformation)

**ENDIF** /\* ENDIF IsPresentPattern(MXRemittanceInformation, "/BNF/") \*/

/\*Search for the /TSU/ codes. It is not expected there will be several ones. \*/

**IF** **IsPresentPattern**(MXRemittanceInformation, "/TSU/") THEN

TempInformation = **ExtractBetweenPattern**(MXRemittanceInformation,”/TSU/”, “/BNF/”)

**IF** TempInformation **NOT IsEmpty** THEN

/\* save information in the table \*/

i = i + 1

Table[i].Code = “/TSU/”

Table[i].Information = TempInformation

**ENDIF**

/\* Delete the code and string extracted \*/

MXRemittanceInformation = **DeletePattern**(MXRemittanceInformation,**Concatenate**(”/TSU/”, TempInformation)

**ENDIF** /\* ENDIF IsPresentPattern(MXRemittanceInformation, "/TSU/") \*/

/\* Check if MXRemittanceInformation has still information which might be the case if the payment is originated in MX. That information will be translated with code /BNF/ \*/

**IF** MXRemittanceInformation **NOT IsEmpty** THEN

/\*concatenate with /BNF/ or create a new element in Table. \*/

**IF**  i > 0 THEN

**For t** = 1 to i

**IF** Table[t].Code = “/BNF/” THEN

Table[t].Information = **Concatenate**(Table[t].Information, “ “, MXRemittanceInformation)

BNFIndicator = “true”

Exit loop For

**ENDIF**

**END Loop**

**ENDIF**

**IF** BNFIndicator = “false” THEN

/\* create a new element in the table \*/

i = i + i

Table[i].Code = “/BNF/

Table[i].Information = MXRemittanceInformation

**ENDIF**

**ENDIF /\*** MXRemittanceInformation NOT IsEmpty \*/

/ \* Fill in Field 72. At this stage at least one line is free and at least one element in the table\*/

/\* Sort the Table to get first code “/BNF/” then “/TSU/” \*/

**For t** = 1 to i

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField

**ELSE**

NumberOfEmptyLines = 6 - **ReturnFirstLineEmpty (MT72, 6)** +1

TempInformation = **Concatenate**(Table[t].Code, Table[t].Information)

**AppendComplexMT72**(NumberOfEmptyLines, TempInformation,MT72; MT72)

**ENDIF**

**END LOOP**

**SubfunctionPreviousInstructingAgent**

/\* Local variables:

MTAgent : string

NumberOfEmptyLines : integer

RequiredLength: integer \*/

**/\* PreviousInstructingAgent1,2,3 \*/**

/\* The same logic applies to each agent. There is a standard rule stating that PreviousInstructingAgent3 cannot be present if PreviousInstructingAgent2 is absent and PreviousInstructingAgent2 cannot be present if PreviousInstructingAgent1 is absent . So assuming that the source message is a valid message as a precondition for translation, there is no need to establish dependencies in the translation rules. \*/

RequiredLength = 210

**IF** PreviousInstructingAgent1 **IsPresent** THEN

MXAgent = MXPreviousInstructingAgent1

**MX\_To\_MTBICNameAgent**(MXAgent, RequiredLength; MTAgent)

**IF** **Length**(MTAgent) > 0 THEN

/ \*check room left \*/

MTAgent = **Concatenate** (“/INS/”, MTAgent)

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ELSE**

/\*Append PreviousInstructingAgent1 to Field 72\*/

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTAgent, MT72;MT72)

**ENDIF**

**ENDIF** /\* IF Length(MTAgent) > 0 \*/

**ENDIF**  /\* PreviousInstructingAgent1 IsPresent \*/

**IF** PreviousInstructingAgent2 **IsPresent** THEN

MXAgent = MXPreviousInstructingAgent2

**MX\_To\_MTBICNameAgent**(MXAgent, RequiredLength; MTAgent)

**IF** **Length**(MTAgent) > 0 THEN

/ \*check room left \*/

MTAgent = **Concatenate** (“/INS/”, MTAgent)

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ELSE** /\*Append PreviousInstructingAgent2 to Field 72\*/

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTAgent, MT72;MT72)

**ENDIF**

**ENDIF** /\* IF Length(MTAgent) > 0 \*/

**ENDIF**  /\* PreviousInstructingAgent2 IsPresent \*/

**IF** PreviousInstructingAgent3 **IsPresent** THEN

MXAgent = MXPreviousInstructingAgent3

**MX\_To\_MTBICNameAgent**(MXAgent, RequiredLength; MTAgent)

**IF** **Length**(MTAgent) > 0 THEN

/ \*check room left \*/

MTAgent = **Concatenate** (“/INS/”, MTAgent)

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ELSE** /\*Append PreviousInstructingAgent3 to Field 72\*/

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTAgent, MT72;MT72)

**ENDIF**

**ENDIF** /\* IF Length(MTAgent) > 0 \*/

**ENDIF**  /\* PreviousInstructingAgent3 IsPresent \*/

**SubfunctionDebtorAgent**

/\* Local variables:

MTAgent : string

NumberOfEmptyLines : integer

RequiredLength: integer

MXAgent : structure typed BranchAndFinancialInstitutionIdentification6 \*/

/\* Same logic as PreviousInstructingAgent translation \*/

RequiredLength = 210

**IF** DebtorAgent **IsPresent** THEN

MXAgent = DebtorAgent

**MX\_To\_MTBICNameAgent**(MXAgent, RequiredLength; MTAgent)

**IF** **Length**(MTAgent) > 0 THEN

/ \*check room left \*/

MTAgent = **Concatenate** (“/INS/”, MTAgent)

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField2

**ELSE**

/\*Append DebtorAgent to Field 72\*/

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTAgent, MT72;MT72)

**ENDIF**

**ENDIF** /\* IF Length(MTAgent) > 0 \*/

**ENDIF**  /\* DebtorAgent IsPresent \*/

### 4**.3.15 MX\_To\_MTAddressLineType**

**Name**

MX\_To\_MTAddressLineType

**Business description**

The function analyses the type of information present in the MX party address line.

If all lines start with “2/” or “3/” then it comes from a previous translation of field 50 F or 59F using a structured address line or from a previous MX with a structured postal address translated to MT and then translated back to MX.

The function checks that :

* each line of AddressLine must start with “2/” or “3/”
* and at least one line starts with “3/”
* and the first line starting with “3/” must be followed by the country code
* and number “2/” cannot follow “3/”
* and max 2 lines with “2/” and max 2 lines with “3/”
* then returns the Boolean StructuredIndicator is true
* else Boolean StructuredIndicator is false.

**Format**

**MX\_To\_MTAddressLineType**(MXParty; StructuredIndicator)

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

**Output**

StructuredIndicator : boolean

**Preconditions**

MX Address Line is present

**None**

**Formal description**

/\* Local variables

Index, i :integer

StructuredIndicator : boolean \*/

StructuredIndicator = “false”

**IF** NumberOfOccurrences(MXParty. PostalAddress.AddressLine)> 0 THEN

StructuredIndicator = “true”

/\* Check each line starts with “2” or “3/” \*/

**For** Index = 1 to NumberOfOccurrences(MXParty. PostalAddress.AddressLine)

**IF** **Substring**(MXParty. PostalAddress.AddressLine[Index], 1,2) NotInList {“2/”, “3/”} Then

StructuredIndicator = “false”

Exit function MX\_To\_MTAddressLineType

**ENDIF**

**END Loop**

/\* Check Number “2/” cannot follow “3/” \*/

**For** Index = 1 to NumberOfOccurrences(MXParty. PostalAddress.AddressLine)

**IF** **Substring**(MXParty. PostalAddress.AddressLine[Index], 1,2) **IsEqual** to “3/” THEN

**i**= Index + 1

**While i <=** NumberOfOccurrences(MXParty. PostalAddress.AddressLine)

**{** **IF** **Substring**(MXParty. PostalAddress.AddressLine[i], 1,2) **Not IsEqual** to “3/” THEN

StructuredIndicator = “false”

Exit function MX\_To\_MTAddressLineType

**ENDIF**

**i = i + 1 }**

**ENDIF**

**END loop**

/\* One line starting with “3/” is mandatory. Check if first line starting with “3/” is followed by a country code. If the country code is followed by additional information it must be preceded by “/” \*/

**For** Index = 1 to NumberOfOccurrences(MXParty. PostalAddress.AddressLine)

**IF** **Substring**(MXParty. PostalAddress.AddressLine[Index], 1,2) **IsEqual** to “3/” THEN

**IF IsValidCountryCode(Substring**(MXParty. PostalAddress.AddressLine[Index], 3,2)**)AND** **[Substring**(MXParty. PostalAddress.AddressLine[Index],3))**has pattern** ([A-Z]{2}|[A-Z]{2}”/” optionally followed by any characters)**]** THEN

StructuredIndicator = “true”

Exit function MX\_To\_MTAddressLineType

**ELSE**

/\* first occurrence of “3/” does not start with country code \*/

StructuredIndicator = “false”

Exit function MX\_To\_MTAddressLineType

**ENDIF /\* Valid country code \*/**

**ENDIF** /\* ENDIF line starts with “/3” \*/

**END LOOP**

/\* No line starting with “/3” is found. So it is not a valid MT structured address. \*/

StructuredIndicator = “false”

/\* check number of lines starting with “2/” \*

i = 0

**For** Index = 1 to NumberOfOccurrences(MXParty. PostalAddress.AddressLine)

**IF** **Substring**(MXParty. PostalAddress.AddressLine[Index], 1,2) **IsEqual** to “2/” THEN

**i = i + 1**

**ENDIF**

**End Loop**

**IF** i > 2 **THEN**

StructuredIndicator = “false”

Exit function MX\_To\_MTAddressLineType

**ENDIF**

/\* check number of lines starting with “3/” \*

i = 0

**For** Index = 1 to NumberOfOccurrences(MXParty. PostalAddress.AddressLine)

**IF** **Substring**(MXParty. PostalAddress.AddressLine[Index], 1,2) **IsEqual** to “3/” THEN

i = i + 1

**ENDIF**

**End Loop**

**IF** i > 2 **THEN**

StructuredIndicator = “false”

Exit function MX\_To\_MTAddressLineType

**ENDIF**

### 4.3.16 MX\_To\_MTStartingLineCharacter

**Name**

MX\_To\_MTStartingLineCharacter

**Business description**

The function replaces by a “.” (dot) the Colon and the Hyphen characters at the beginning of each line starting with the second line in multiline fields (eg., first line without FIN tag)

If the lines start with multiple consecutive hyphen or colon characters, only the first one is replaced by the dot.

**Format**

**MX\_To\_MTStartingLineCharacter**(InMultilineField; OutMultilineField)

**Input**

InMultilineField is a MT multiline Fields\*.

**Output**

OutMultilineField is a MT multiline Fields\*.

(\*) Multiline Fields being for example in MT103 Field 70, Field 72, Field 77, 52D, 53D, 54D, 55D, 56D, 57D, 50K, 59 No letter.

**Preconditions**

**None**

**Formal description**

/\* Local variables

NumberOfLines, i, j, : number

Replacementchar : string

NotAllowedPattern :Pattern (regular expression)

, \*/

/\* InMultilineField[n] and OutMultilineField[n] designate the line n of the multiline field. \*/

/\* The line 2 and next lines cannot start neither with Colon = “:” nor Hyphen = “-“ \*/

ReplacementChar = “.”

NotAllowedPattern = [Hyphen Colon]

/\* Extract the number of lines not empty from InMultilineField. In case the field has subfield1 and subfield2 (eg 50K), the total number of lines under the same FIN tag is returned \*/

NumberOfLines = **NumberOfOccurrences**(InMultilineField)

/\* assumption is that in the input, empty lines cannot be present between used lines \*/

**IF NumberOfLines** <= 1THEN

Exit function

/\* If NumberOfLines less or equal to 1, nothing to remove \*/

**ENDIF**

**For** i = 2 to NumberOfLines

**IF** **IsPresentPattern**(**Substring**(InMultilineField[i],1,1) NotAllowedPattern)THEN

OutMultilineField[i] = **ReplacePattern**(**Substring**(InMultilineField[i],1,1)NotAllowedPattern,ReplacementChar)

**ENDIF**

**Next** i

### 4.3.17 MX\_To\_MTRemittanceInformation2

**Name**

MX\_To\_MTRemittanceInformation2

**Business description**

The function translates UltimateDebtor, UltimateCreditor information to field 70 Remittance information from pacs.004 to MT103

The UltimateParties are extracted from the ReturnChain component.

The following code words will be used /ULTB/ to designate the ultimate creditor in the return and /ULTD/ to designate the ultimate debtor in the return.

/ULTB/ is followed by information about the UltimateCreditor BIC or (Name/Country [/TownName]), TownName is optional or (Name/OtherId) or Name alone or OtherId alone. Depending on the information available, that priority will be applied

/ULTD/ is followed by information about the UltimateDebtor BIC or (Name/Country/TownName), TownName is mandatory or (Name/OtherId) or Name alone or OtherId alone. Depending on the information available, that priority will be applied

IF BIC is present for the ultimate party, it is translated and the other informations are ignored.

The MTRemittanceInformation is built applying **priority** in the information ie., UltimateCreditor is first translated follows by UltimateDebtor. This means that the information can be truncated (eg., if the Name of ultimate creditor is about 140 characters) . Sign “+” is added at the end of the string to indicate characters truncation. If a **full data** is not copied because lack of room then the indicator Flag\_MissingInformation is returned with value “True””.

**Format**

MX\_To\_MTRemittanceInformation2 (MXReturnUltimateCreditor, MXReturnUltimateDebtor; MTRemittanceInfo)

**Input**

MXReturnUltimateCreditor, MXReturnUltimateDebtor typed PartyIdentification135

MXReturnUltimateCreditor and MXReturnUltimateDebtor are located in ReturnChain component.

**Output**

MTRemittanceInfo: remittance information in an MT field 70 format of 4\*35x.

**Preconditions**

At least MXReturnUltimateCreditor or MXReturnUltimateDebtor should not be empty

**Formal description**

/\* Local variables

UCIndicator, UDIndicator : Boolean

MTUltimateCreditor, MTUltimateDebtor, MT70FullString : string \*/

**IF** MXReturnUltimateCreditor AND MXReturnUltimateDebtor **IsEmpty** THEN

EXIT Function

/\* No Remittance information field \*/

**ENDIF**

/\* Fill in information from Ultimate Parties \*/

UCIndicator = “false”

UDIndicator = “false”

**IF** MXReturnUltimateCreditor **IsPresent**  THEN

**MX\_To\_MTUltimateParty**(MXReturnUltimateCreditor, MTUltimateCreditor)

**IF** **Length**(MTUltimateCreditor) > 0 THEN

UCIndicator = “True”

MTUltimateCreditor = **Concatenate**(“”/ULTB/”, MTUltimateCreditor)

**IF** **Length** (MTUltimateCreditor) > 140 THEN

MT70FullString = **Concatenate**(**Substring**(MTUltimateCreditor,1, 139),”+”))

**ELSE**

MT70FullString = **Substring**(MTUltimateCreditor,1, **Length**(MTUltimateCreditor))

**ENDIF**

**ENDIF** /\* IF Length(MTULtimateCreditor) > 0 \*/

**ENDIF** /\* MXReturnUltimateCreditor IsPresent \*/

**IF** MXReturnUltimateDebtor **IsPresent** THEN

**MX\_To\_MTUltimateParty** (MXReturnUltimateDebtor, MTUltimateDebtor)

**IF** **Length**(MTUltimateDebtor) > 0 THEN

{ UDIndicator = “True”

**CASE UCIndicator = “True”**

/\* Next Concatenation starting with “///ULTD/”, \*/

RemaingLine = 140 – **Length**(MT70FullString) - 8

**IF** RemainingLine > 0 THEN

**IF** **Length**(MTUltimateDebtor) > RemainingLine THEN

MT70FullString = **Concatenate**(MT70FullString, “///ULTD/”,**Substring**(MTUltimateDebtor,1,RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**(MT70FullString, “///ULTD/”, **Substring**(MTUltimateDebtor, 1))

**ENDIF**

**ELSE**

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

**END CASE** /\* **UCIndicator = “True” \*/**

**CASE UCIndicator = “False ”**

/\* Next Concatenation starting with “/ULTD/”, \*/

RemaingLine = 140 – **Length**(MT70FullString) - 6

**IF** RemainingLine > 0 THEN

**IF** **Length**(MTUltimateDebtor) > RemainingLine THEN

MT70FullString = **Concatenate**( “/ULTD/”, **Substring**(MTUltimateDebtor,1,RemainingLine-1), “+”)

**ELSE**

MT70FullString = **Concatenate**( “/ULTD/”, **Substring**(MTUltimateDebtor, 1))

**ENDIF**

**ELSE /\*** RemainingLine = 0 \*/

/\*no partial data can be copied \*/

Flag\_MissingInformation = “True”

**ENDIF**

}

**ENDIF** /\* LENGTH(MTUltimateDebtor) > 0 \*/

**ENDIF** /\* MXReturnUltimateDebtor IsPresent \*/

/\* Copy the MT70FullString in 4\*35 Format \*/

**IF** **Length**(MT70FullString)> 105 THEN

MTRemittanceInfo[1] = **Substring**(MT70FullString,1,35)

MTRemittanceInfo[2] = **Concatenate**(CRLF,**Substring**(MT70FullString,36,70)

MTRemittanceInfo[3] = **Concatenate**(CRLF,**Substring**(MT70FullString,71,105)

MTRemittanceInfo[4] = **Concatenate**(CRLF,**Substring**(MT70FullString, 106)

**ELSEIF** **Length**(MT70FullString)> 70 THEN

MTRemittanceInfo[1] = **Substring**(MT70FullString,1,35)

MTRemittanceInfo[2] = **Concatenate**(CRLF,**Substring**(MT70FullString,36,70)

MTRemittanceInfo[3] = **Concatenate**(CRLF,**Substring**(MT70FullString, 71)

**ELSEIF** **Length**(MT70FullString)> 35 THEN

MTRemittanceInfo[1] = **Substring**(MT70FullString, 1, 35)

MTRemittanceInfo[2] = **Concatenate**(CRLF,**Substring**(MT70FullString, 36)

**ELSEIF** LENGTH(MT70FullString)<= 35

MTRemittanceInfo[1] = **Substring(**MT70FullString,1)

**ENDIF**

### 4.3.18 MX\_To\_MT72FullField3

**Name**

MX\_To\_MT72FullField3

**Business description**

The function builds the field 72 used for a returned payment using a new MT103/MT202 as target message and pacs.004 as a source message.

The field 72 will contain information about the return reason and possibly information from original payment Local Instrument (code only), IntermediaryAgent2,3 and PreviousInstructionAgent1,2,3 if present in the return chain and if room left in field 72.

If the target message is MT202, additional information about Charge Bearer and charge amount(s) are translated to Field 72 as such without any calculation like for field 71G in MT103 and without any dependency between the presence of ChargeBearer and ChargesInformation, meaning that ChargesInformation may be present without ChargeBearer in pacs.004.

The following code words and priority apply:

**IF Target message is MT103 :**

Priority 1 : Return Reason Code, /RRC/

Priority 2 : IntermediaryAgent2,3, /INTA/

Priority 3 : Local Instrument /LOCINS/Code

Priority 4 : PreviousInstructingAgent1,2,3, /INS/

**IF Target Message is MT202 :**

Priority 1 : Return Reason Code, /RRC/

Priority 2 : IntermediaryAgent2,3, /INTA/

Priority 3 : Charge Bearer /CHGB/ and Amounts /CHGA/ which can be repeated.

Priority 4 : Local Instrument /LOCINS/Code

Priority 5 : PreviousInstructingAgent1,2,3, /INS/

The Reason code line(s) will have the following structure:

If the MX reason Code has an equivalent (code and code meaning) MT reason code:

:72:/RRC/MTErrorCode/Text1

//Text2

//Text3

If the MX reason Code has no equivalent MT reason code (code and code meaning):

:72:/RRC/XT99/ISOErrorCode/Text1

//Text2

//Text3

Where XT99 is a fixed code meaning that there is no MT error code equivalent and the next information is the MX ISO code.

Text1, Text2, Text3, … is from the MX Additional information split in multiple lines if needed.

If a data (string) has to be truncated, a sign “+” is added at the end of the data. If all the information cannot be translated to field 72, the Flag\_Missinginformation will get the value “true”.

**Format**

MX\_To\_MT72FullField3 (MXReturnReasonInformation, MXReturnChain, MXOriginalTransactionReference, MXChargeBearer, MXChargesInformation; MT72)

**Input**

MXReturnReasonInformation : MX message element typed PaymentReturnReason6

MXReturnChain : MX message element typed TransactionParties7

MXOriginalTransactionReference : MX message element typed OriginalTransactionReference28

MXChargeBearer : MX element typed ChargeBearerType1Code

MXChargesInformation : MX message element typed Charges7

**Output**

MT72 : field with structure of field 72

**Preconditions**

None

**Formal description**

**Call SubfunctionReturnReason(**MXReturnReasonInformation, MT72; MT72)

/\*SubfunctionReturnReason described below \*/

**IF** MXReturnChain/IntermediaryAgent2 **IsPresent** THEN

**Call** **SubfunctionIntermediaryAgents(**MXReturnChain,MT72;MT72)

/\*For developers only : **SubfunctionIntermediaryAgents** described in MX\_To\_MT72FullField2 but replace

“Exit Function MX\_To\_MT72FullField2” by Exit Function “MX\_To\_MT72FullField3”

IntermediaryAgent2,3 are extracted from MXReturnChain element \*/

**ENDIF**

/\* Charges information to be translated to Field 72 only if the target message is MT202 \*/

**IF Target message is “MT202” AND IF (**MXChargeBearer ORMXChargesInformation) **IsPresent** THEN

**Call** **SubfunctionChargesInformation(**MXChargeBearer, MXChargesInformation, MT72; MT72)

/\* SubfunctionChargesInformation described below \*/

**ENDIF**

**IF** MXOriginalTransactionReference /PaymentTypeInformation/LocalInstrument/Code **IsPresent AND NOT** **InList**{CRED,CRTS,SPAY,SPRI, SSTD} THEN

**Call SubfunctionLocalInstrument1(**MXOriginalTransactionReference /PaymentTypeInformation/LocalInstrument/Code, MT72; MT72)

**ENDIF**

**IF** MXReturnChain /PreviousInstructingAgent1 **IsPresent** THEN

**Call** **SubfunctionPreviousInstructingAgent(**MXReturnChain, MT72; MT72)

/\*For developers only : **SubfunctionPreviousInstructingAgent** described in MX\_To\_MT72FullField2 but replace

“Exit Function MX\_To\_MT72FullField2” by Exit Function “MX\_To\_MT72FullField3”

PreviousInstructingAgent1,2,3 are extracted from MXReturnChain element

\*/

**ENDIF**

**/\* Subfunctions description \*/**

**SubfunctionReturnReason**

/\* The table with MT MX reason code equivalence is described in the Excel file pacs.004 to MT103 New, sheet “Error Codes RETN” \*/

/\* Local variables

MXReturnCode :string ([A-Z0-9]{4})

MTReturnCode, MTFixedCode :string([A-Z0-9]{4})

MTReturnReasonLine : string

MXAdditionalInformation : string

ErrorCodeTable : Excel pacs.004 to MT103 New, sheet “Error Codes RETN”

IsMTErrorCodePresent: boolean

MTReturnCodeWord : string \*/

MTReturnCodeWord = “/RRC/”

MTFixedCode = “XT99”

MXReturnCode = MXReturnReasonInformation.Reason.Code

**IF** **Length**(MXReturnReasonInformation.AdditionalInformation[1]) > 0 THEN

MXAdditionalInformation = MXReturnReasonInformation.AdditionalInformation[1]

**IF** **Length**(MXReturnReasonInformation.AdditionalInformation[2]) > 0 THEN

**IF** **Length**(MXReturnReasonInformation.AdditionalInformation[1]) > 104

/\* Assumption is that the next line is the continuation of the first one and therefore no space is added between the 2 occurrences \*/

THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXReturnReasonInformation.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXReturnReasonInformation.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF**

/\* In “Error Codes RETN” sheet in EXCEL document describing pacs.004 to MT103 New, col H, search for MXReturnCode. Return the corresponding value IsMTErrorCodePresent from column D. If the MXReturnCode is not found in col H (ie., excel list not up to date), IsMTErrorCodePresent = “False”.

If IsMTErrorCodePresent is “True”, extract the MT equivalent code named MTReturnCode from column A \*/

**IF** **IsInTable**(MXReturnCode) THEN

**IF** **IsMTErrorCodePresent** THEN

MTReturnCode = **GetEquivalent**(MXReturnCode)

MTReturnReasonLine = **Concatenate**(MTReturnCodeWord,MTReturnCode,”/”, MXAdditionalInformation)

**ELSE**

/\* No MT equivalent code \*/

MTReturnReasonLine = **Concatenate**(MTReturnCodeWord,MTFixedCode,”/”,MXReturnCode,”/”, MXAdditionalInformation)

**ENDIF**

**ELSE**

/\* MX code is not in the table but must still be a valid ISO code. This assumption is to avoid maintenance issue with the table in Excel \*/

MTReturnReasonLine = **Concatenate**(MTReturnCodeWord,MTFixedCode,”/”,MXReturnCode,”/”, MXAdditionalInformation)

**ENDIF**

/\* Append to field MT72 \*/

**IF** Length(MTReturnReasonLine) > 0 THEN

/\* Check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField3

**ENDIF**

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTReturnReasonLine, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** Length(MTReturnReasonLine) > 0 \*/

/\* END **SubfunctionReturnReason** \*/

**SubfunctionLocalInstrument1(**MXLocalInstrument, MT72; MT72)

**IF** **Length**(MXLocalInstrument) > 0 THEN

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField3

**ENDIF**

MXLocalInstrument = **Concatenate** (“/LOCINS/”, MXLocalInstrument)

**NumberOfEmptyLines = 6 – ReturnFirstLineEmpty (MT72, 6) + 1**

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MXLocalInstrument, MT72;MT72)

**ELSE**

/\* nothing to translate \*/

**ENDIF /\*** Length(MXLocalInstrument) > 0 \*/

**SubfunctionChargesInformation(**MXChargeBearer, MXChargesInformation, MT72; MT72)

/\* Local variables

MXChargeAmount : Amount

MTChargeBearer : code in list {BEN, OUR, SHA}

MTAmount : MT Amount type

MTCurrency : ISO currency code

MTCurrencyAmount : string

MTChargeBearerCodeWord : string

MTChargeAmountCodeWord : string \*/

MTChargeBearerCodeWord = “/CHGB/”

MTChargeAmountCodeWord = “/CHGA/”

/\* SLEV is not allowed \*/

**IF** MXChargeBearer = “CRED” THEN

MTChargeBearer = “BEN”

**ELSEIF** MXChargeBearer = “DEBT” THEN

MTChargeBearer = “OUR”

**ELSEIF** MXChargeBearer = “SHAR” THEN

MTChargeBearer = “SHA”

**ENDIF**

**IF** **Length**(MTChargeBearer) > 0 THEN

MTChargeBearer = **Concatenate**(MTChargeBearerCodeWord, MTChargeBearer)

/ \*check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField3

**ENDIF**

**NumberOfEmptyLines = 6 – ReturnFirstLineEmpty (MT72, 6) + 1**

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTChargeBearer, MT72;MT72)

**ENDIF**

**IF** MXChargesInformation **IsPresent** THEN

**For i** = 1 to **NumberOfOccurrences**(MXChargesInformation) THEN

MXChargeAmount = MXChargesInformation[i].Amount

**MX\_To\_MTCurrencyAmount**(MXChargeAmount; MTCurrency, MTAmount)

MTCurrencyAmount = **Concatenate**(MTChargeAmountCodeWord, MTCurrency, MTAmount)

/ \*check room left \*/

**IF ReturnFirstLineEmpty**(MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### Exit Function MX\_To\_MT72FullField3

**ENDIF**

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTCurrencyAmount, MT72;MT72)

**Next i**

**ENDIF**  /\* MXChargesInformation IsPresent \*/

### 4.3.19 MX\_To\_MTRegulatoryReporting

**Name**

MX\_To\_MTRegulatoryReporting

**Business description**

MX RegulatoryReporting/Details/Information is translated to field 77B in a pure textual format in order to save space (meaning that if the code word “/ORDERRES/” or “/BENEFRES/” is present in MX RegulatoryReporting, it will not be extracted from MX to start a new line in MT Field 77B.

If there is still room left, and if the code words “/ORDERRES/”, “/BENEFRES/” are not present in MX RegulatoryReporting/Details/Information, the function extracts information from Debtor and Creditor CountryOfResidence, if present and concatenates with the MT code word respectively /ORDERRES/ and /BENEFRES/. This information is concatenated to field 77B, if there is still room left.

It is assumed that if “/ORDERRES/” or “/BENEFRES/” if present in MX RegulatoryReporting/Details/Information, is followed with the country of residence of the debtor, creditor respectively. But no check is done. If the Debtor (Creditor) country of residence is also present, it will be ignored and no missing information is reported.

If truncation is needed, sign “+” is added.

Other data present in the component RegulatoryReporting will not be translated.

**Format**

**MX\_To\_MTRegulatoryReporting**(MXRegulatoryReporting, Debtor, Creditor; MT77)

**Input**

MXRegulatoryReporting typed RegulatoryReporting3

Debtor, Creditor typed PartyIdentifier135

**Output**

MT77 is field MT 77B (3\*35 char)

**Preconditions**

None

**Formal description**

/\* Local variables

NumberOfRegulatoryReportingOccurrences, NumberOfDetailsOccurrences, NumberOfInformationOccurrences, RemainingSpace : integer

DebtorCountryOfResidenceIndicator, CreditorCountryOfResidenceIndicator, BENEFFound,ORDERFound : Boolean

MTRegulatoryReportingConcatenated, BENEFCode, ORDERCode : string \*/

DebtorCountryOfResidenceIndicator = false

CreditorCountryOfResidenceIndicator = false

BENEFFound = false

ORDERFound = false

BENEFCode = “/BENEFRES/”

ORDERCode = “/ORDERRES/”

MTRegulatoryReportingConcatenated = “”

**IF** RegulatoryReporting/Details/Information **IsPresent** THEN

MTRegulatoryReportingConcatenated **= Subfunction** **ExtractMTRegulatoryReportingInformation(**MXRegulatoryReporting**;** MTRegulatoryReportingConcatenated)

/\* Subfunctions described below. \*/

/\* remove space(s) at the end possibly added in the last occurrence. Consequence of the Trimright function is that all spaces on the right, even the ones belonging to the data are removed. It is not the principle applied in other data translation. If this is an issue, then only the last SPACE character will be removed, knowing that there is still a risk it is part of the data in case the last line consumes 35 char with a SPACE as last character. \*/

MTRegulatoryReportingConcatenated = **TrimRight**(MTRegulatoryReporting, SPACE)

**ENDIF** /\* RegulatoryReporting/Details/Information IsPresent \*/

/\* Concatenate with Debtor/Creditor CountryOfResidence, if not present in RegulatoryReporting and if room left. At least one character after the code word “/ORDERRES/” or “/BENEFRES/” can be copied otherwise CountryOfResidence is reported as missing \*/

**IF** Debtor.CountryOfResidence IsPresent THEN

DebtorCountryOfResidenceIndicator = true

**ENDIF**

**IF** Creditor.CountryOfResidence IsPresent THEN

CreditorCountryOfResidenceIndicator = true

**ENDIF**

**IF** IsPresentPattern(MTRegulatoryReportingConcatenated, BENEFCode) THEN

BENEFFound = true

**ENDIF**

**IF** IsPresentPattern(MTRegulatoryReportingConcatenated, ORDERCode) THEN

ORDERFound = true

**ENDIF**

**IF** ORDERFound = False AND DebtorCountryOfResidenceIndicator = True THEN

/\* this case handles also the case where RegulatoryReporting is absent because ORDERFound = false if MTRegulatoryReportingConcatenated is empty \*/

RemainingSpace = 105 – **Length**(MTRegulatoryReportingConcatenated) – **Length**(ORDERCode)

**IF** RemainingSpace > 0 THEN

**IF** **Length**(Debtor.CountryOfResidence) > RemainingSpace THEN

MTRegulatoryReportingConcatenated = **Concatenate**(MTRegulatoryReportingConcatenated, ORDERCode, **Substring**(Debtor.CountryOfResidence, 1, RemainingSpace – 1) , “+”)

**ELSE**

MTRegulatoryReportingConcatenated = **Concatenate**(MTRegulatoryReportingConcatenated, ORDERCode, Debtor.CountryOfResidence)

**ENDIF**

**ELSE**

/\* Debtor Country of Residence is missing \*/

Flag\_MissingInformation = “true”

**ENDIF** /\* End IF RemainingSpace > 0 \*/

**ENDIF** /\* ENDIF ORDERFound \*/

**IF** BENEFFound = False AND CreditorCountryOfResidenceIndicator = True THEN

RemainingSpace = 105 – **Length**(MTRegulatoryReportingConcatenated) – **Length**(BENEFCode)

**IF** RemainingSpace > 0 THEN

**IF** **Length**(Creditor.CountryOfResidence) > RemainingSpace THEN

MTRegulatoryReportingConcatenated = **Concatenate**(MTRegulatoryReportingConcatenated, BENEFCode, **Substring**(Creditor.CountryOfResidence, 1, RemainingSpace – 1) , “+”)

**ELSE**

MTRegulatoryReportingConcatenated = Concatenate(MTRegulatoryReportingConcatenated, BENEFCode, Creditor.CountryOfResidence)

**ENDIF**

**ELSE**

/\* Creditor Country of Residence is missing \*/

Flag\_MissingInformation = “true”

**ENDIF** /\* ENDIF RemainingSpace > 0 \*/

**ENDIF** /\* ENDIF BENEFFound \*/

/\* Copy information to Field 77B \*/

**IF** **Length**(MTRegulatoryReportingConcatenated) > 0 THEN

**Call** **Subfunction FillInMTRegulatoryReportingInformation (**MTRegulatoryReportingConcatenated; MT77)

**ENDIF**

**/\* Subfunction specifications \*/**

**Subfunction** **ExtractMTRegulatoryReportingInformation(**MXRegulatoryReporting**;** MTRegulatoryReporting)

/\* Extract the information from RegulatoryReporting/Details/Information and build a string by concatenation of the occurrences \*/

/\* Local variables

MXRegulatoryReporting typed RegulatoryReporting3

MTRegulatoryReporting : string

i,j,k : integer

NumberOfRegulatoryReportingOccurrences, NumberOfDetailsOccurrences, NumberOfInformationOccurrences : integer \*/

NumberOfRegulatoryReportingOccurrences = **NumberOfOccurences**(RegulatoryReporting)

**IF** NumberOfRegulatoryReportingOccurrences > 0 THEN

**For** **i** = 1 to NumberOfRegulatoryReportingOccurrences

NumberOfDetailsOccurrences = **NumberOfOccurences**(RegulatoryReporting[i]/Details)

**IF** NumberOfDetailsOccurrences > 0 THEN

**For** j = 1 to NumberOfDetailsOccurrences

NumberOfInformationOccurrences = **NumberOfOccurences**(RegulatoryReporting[i]/Details[j]/Information)

**IF** NumberOfInformationOccurrences > 0 THEN

**For** k = 1 to NumberOfInformationOccurrences

/\* If the line is not full, add a space \*/

**IF** **Length**(RegulatoryReporting[i]/Details[j]/Information[k]) < 35 THEN

RegulatoryReporting[i]/Details[j]/Information[k] = **Concatenate**(RegulatoryReporting[i]/Details[j]/Information[k], SPACE)

**ENDIF**

MTRegulatoryReporting = **Concatenate**(MTRegulatoryReporting, RegulatoryReporting[i]/Details[j]/Information[k])

**Next k**

**ENDIF**

**Next j**

**ENDIF**

**Next i**

**ENDIF**

**Subfunction FillInMTRegulatoryReportingInformation(MTRegulatoryReporting; MT77)**

/\* Copy the MTRegulatoryReporting information to field 77B by just splitting over 3 lines \*/

/\* Local variables

MTRegulatoryReporting : string

MT77 : Field with MT 77B structure \*/

**IF Length**(MTRegulatoryReporting) < 1 THEN

Exit Subfunction

**ENDIF**

**IF** **Length**(MTRegulatoryReporting) > 105 THEN

MTRegulatoryReporting = **Concatenate**(**Substring**(MTRegulatoryReporting,1, 104), “+”)

**ENDIF**

/\* Copy to MT77 \*/

**IF** **Length**(MTRegulatoryReporting) < 36 THEN

**AppendToNextLine**(Substring(MTRegulatoryReporting,1,35), MT77)

**ELSEIF Length**(MTRegulatoryReporting) < 71 THEN

**AppendToNextLine**(Substring(MTRegulatoryReporting,1,35), MT77)

**AppendToNextLine**(Substring(MTRegulatoryReporting,36,35), MT77)

**ELSEIF Length**(MTRegulatoryReporting) < 106 THEN

**AppendToNextLine**(Substring(MTRegulatoryReporting,1,35), MT77)

**AppendToNextLine**(Substring(MTRegulatoryReporting,36,35), MT77)

**AppendToNextLine**(Substring(MTRegulatoryReporting,71,35), MT77)

**ENDIF**

### 4.3.20 MX\_To\_MT79FullField

**Name**

MX\_To\_MT79FullField

**Business description**

The function builds the field 79 used for a rejected payment using a MT199/MT299 as target message and pacs.002 as a source message.

The field 79 will contain information about the reject reason and references identifying the payment which is rejected.

The first line of Field 79 must start with the code word /RJCT/ defining the scope of the MT199/MT299.

The following code words and priority apply:

Priority 1 : Reject status Code, /RJCT/

Priority 2 : Original UETR, /UETR/

Priority 3 : Original Message Identification, /MREF/

Priority 4 : Reject Reason Information, /RRC/

Priority 5 : Original End To End Identification, /ROC/

Priority 6 : Clearing System Reference, /CLRF/

The Reject Reason information line(s) will have the following structure:

/RRC/MXRejectReasonCodeorProprietary/AdditionalInformation

AdditionalInformation (continued)

AdditionalInformation (continued

…

AdditionalInformation from the MX Additional information split in multiple lines if needed.

Each code word will start a new line.

**Format**

MX\_To\_MT79FullField (MXTransactionInformationandStatus; MT79)

**Input**

MXTransactionInformationAndStatus : MX message element typed PaymentTransaction110

**Output**

MT79 : field with structure of field 79 (35 lines of 50 characters)

**Preconditions**

None

**Formal description**

/\* Local variables

MXUETR, MXOriginalMsgId, MXOriginalMsgName, MXRejectReason, MXAdditionalInformation, , MXOriginalE2EID, MXClearingSystemReference : string

MTRejectCodeword, MTUETRCodeword, MTOriginalMsgIdCodeword, MTOriginalMsgNameCodeword, MTRejectInfoCodeword, MTOriginalE2EIDCodeword, MTClearingSystemCodeword : string

MTRejectReason : string

NumberLinesNeeded, i : integer \*/

MTRejectCodeword = “/RJCT/”

MTUETRCodeword = “/UETR/”

MTOriginalMsgIdCodeword = “/MREF/”

MTRejectInfoCodeword = “/RRC/”

MTOriginalE2EIDCodeword = “/ROC/”

MTClearingSystemCodeword = “/CLRF/”

/\* Extraction of MX information and format transformation if needed \*/

MXUETR = MXTransactionInformationAndStatus.OriginalUETR

MXOriginalMsgId = MXTransactionInformationAndStatus.OriginalGroupInformation.OriginalMessageIdentification

MXOriginalE2EID = MXTransactionInformationAndStatus.OriginalendToEndIdentification

MXClearingSystemReference = MXTransactionInformationAndStatus.ClearingSystemReference

**IF** MXTransactionInformationAndStatus.StatusReasonInformation.Reason.Code **IsPresent** THEN

MXRejectReason = MXTransactionInformationAndStatus.StatusReasonInformation.Reason.Code

**ELSEIF** MXTransactionInformationAndStatus.StatusReasonInformation.Reason.Proprietary **IsPresent** THEN

MXRejectReason = MXTransactionInformationAndStatus.StatusReasonInformation.Reason.Proprietary

**ENDIF**

**IF** MXTransactionInformationAndStatus.StatusReasonInformation.AdditionalInformation **IsPresent** THEN

MXAdditionalInformation = MXTransactionInformationAndStatus.StatusReasonInformation.AdditionalInformation[1]

**IF** MXTransactionInformationAndStatus.StatusReasonInformation.AdditionalInformation[2] **IsPresent** THEN

**IF** **Length**(MXAdditionalInformation) = 105 THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXTransactionInformationAndStatus.StatusReasonInformation.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXTransactionInformationAndStatus.StatusReasonInformation.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF** /\* AdditionalInformation IsPresent \*/

/\* create the MT string containing reject reason and additional information that can only be present if reject reason is “NARR” as per rule in MX. No double check in translation as it is assumed that the source message is valid \*/

**IF LENGTH**(MXRejectReason) > 0 Then

MTRejectReason = MXRejectReason

**IF** **Length**(MXAdditionalInformation) > 0 THEN

MTRejectReason = **Concatenate**(MTRejectReason, “/”, MXAdditionalInformation)

**ENDIF**

**ENDIF**

/\* Fill in Fill 79 with codewords defined above – 35 lines of 50 characters are available \*/

/\* First line gets the reject status code word and other lines are filled as per priority above \*/

**AppendToNextLine**(MTRejectCodeword, MT79)

**AppendToNextLine**(Concatenate(MTUETRCodeword,MXUETR), MT79)

**AppendToNextLine**(Concatenate(MTOriginalMsgIdCodeword,MXOriginalMsgId), MT79)

/\* Max 6 lines of 50 char are needed for MTRejectReason information \*/

**IF** **Length**(MTRejectReason) > 0 THEN

MTRejectReason = **Concatenate**(MTRejectInfoCodeword, MTRejectReason)

NumberLinesNeeded = **Integer**(**Length**(MTRejectReason)/50) + 1

/\* Split MTRejectReason in multiple lines of max 50 characters \*/

**For i** = 1 to NumberLinesNeeded

**AppendToNextLine**(**Substring**(MTRejectReason, (i-1)\*50+1, 50), MT79)

**Next i**

**ENDIF**

**AppendToNextLine**(Concatenate(MTOriginalE2EIDCodeword,MXOriginalE2EID), MT79)

**IF** **Length**(MXClearingSystemReference) > 0 THEN

**AppendToNextLine**(**Concatenate**(MTClearingSystemCodeword,MXClearingSystemReference), MT79)

**ENDIF**

### 4.3.21 MX\_To\_MT72RETN

**Name**

MX\_To\_MT72RETN

**Business description**

The function builds the field 72 used in a return p ayment (eg MT103 RETN, MT202 RETN or MT205 RETN) when translated from pacs.004 as a source message.

Structure of field 72 is as follows:

-Line 1 : /RETN/2!n where 2!n is the tag of the field in error. Default value “99” is used as identification of the tag in error from pacs.004 is not straightforward.

-Line 2 : /2!c2!n/[29x] contains the MT return reason code follows by narrative information. The MT code is obtained by conversion from MX code. If no equivalent exists, the default value /XT99/ is used follows by the MX ISO Error code, ie, /XT99/ISOErrorCode/[Text1]

IF 29 characters are not sufficient to copy the textual information then it will be continued in Line 6.

-Line 3 : /MREF/16x contains field 20 of the original message (ie payment message)

-Line 4 : /TREF/16x contains the transaction reference

-Line 5 in MT103 : /CHGS/ is not used as Charges are translated to field 71F or 71G using another function

-Line 5 in MT202 205 : /CHGS/ is used to contain the total of MX Charge Amounts

-Line 6 : /TEXT/29x : narrative description that can be continued on next line starting with “//”

The function also handles the translation of the specific structure with MX Reason code is “NARR”

And MX Additional Information has the structure “MTReasonCode[/AdditionalInformation]” which comes from a previous translation MT to MX when the MTReasonCode has no MX equivalent (refer to function MT\_To\_MXReturn72).

**Format**

MX\_To\_MT72RETN (MXReturnReasonInformation, MXTransactionInformation, MTType; MT72)

**Input**

MXReturnReasonInformation: MX message element typed PaymentReturnReason6

MXTransactionInformation: MX message element typed PaymentTransaction112

MTType : type of the output message {MT1,MT2} where MT1 means category 1 and MT2 means category 2

**Output**

MT72 : field with structure of field 72 (6 lines of 35 characters, see MT103 /RETN/)

**Preconditions**

None

**Formal description**

/\* Local variables

MXReasonCode, MXAdditionalInformation, NarrativeLeft, MTReasonCode: string

MTLine[]: table of string

IsInstructionID, IsNARRSpecificScenario : Boolean

NumberOfEmptyLines : integer

\*/

/\* Extract MX information \*/

MXReasonCode = MXReturnReasonInformation.Reason.Code

**IF** **Length**(MXReturnReasonInformation.AdditionalInformation[1]) > 0 THEN

MXAdditionalInformation = MXReturnReasonInformation.AdditionalInformation[1]

**IF** **Length**(MXReturnReasonInformation.AdditionalInformation[2]) > 0 THEN

**IF** **Length**(MXReturnReasonInformation.AdditionalInformation[1]) > 104

/\* Assumption is that the next line is the continuation of the first one and therefore no space is added between the 2 occurrences \*/

THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXReturnReasonInformation.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXReturnReasonInformation.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF**

**/\* Buile MTLine 1 \*/**

MTLine[1] = **Concatenate**(“/RETN/”, “99”)

**/\* Build MTLine 3 \*/**

IsInstructionID = “true”

**Call SubfunctionIDTruncation**(MXtransactionInformation.OriginalInstructionID, IsInstructionID; MTID)

MTLine[3] = **Concatenate**(“/MREF/”, MTID)

**/\* Build MTLine 4 - optional \*/**

**IF Length**(MXTransactionInformation.OriginalEndToEndID)> 0 and

MXtransactionInformation.OriginalEndToEndID **NOT Equal** to “NOTPROVIDED” THEN

{ IsInstructionID = “false”

**Call SubfunctionIDTruncation**(MXtransactionInformation.OriginalEndToEndID, IsInstructionID; MTID)

MTLine[4] = **Concatenate**(“/TREF/”, MTID)}

**ELSE** MTLine[4] = “”

**ENDIF**

**/\* Build MTLine 5 /CHGS/ only if Output message is MT202/MT205 \*/**

MTLine[5] **= “”**

/\* Check if MTtype is MT103 or MT202/MT205 \*/

**IF** MTType = “MT2” THEN

**Call SubfunbctionChargeCalculation(**MXTransactionInformation;MTLine[5])

**ENDIF** /\* MTType = “MT202/205” \*/

**/\* Build MTLine 2 and MTLine 6 \*/**

IsNARRSpecificScenario = “False”

**IF** MXReasonCode = “NARR” AND **Length**(MXAdditionalInformation) > 0 THEN

/\* IF NARR is present then AdditionalInformation is mandatory \*/

**Call** **SubfunctionNARRSpecificScenario**(MXAdditionalInformation;IsNARRSpecificScenario, MTLine[2], MXAdditionalInformation)

/\* The subfunction checks if MXAdditionalInformation has a structure like “MTReasonCode[/AdditionalInformation]” and if MTReasonCode is in the list of allowed codes in MT. if yes, then the MT code is extracted and fills Line[2] and removed from MXAdditionalInformation. In that case,IsNARRSpecificScenario gets value “true”. If not, IsNARRSpecificScenario gets value “false” and MXAdditionalInformation is unchanged \*/

**ENDIF**

**IF NOT** IsNARRSpecificScenario THEN

/\* Cases where IsNARRSpecificScenario = “False” : cases where NARR is present and MXAdditionalInforation is pure narrative (ie no MTReasonCode) or MXCode is different from “NARR” \*/

**Call SubfunbctionReasonCodeTranslation** (MXReasonCode; MTLine[2])

/\* SubfunbctionReasonCodeTranslation is described below. Line[2] is returned with the format “/MTReasonCode/” OR if the MXCode is NARR or has no MT equivalent,the structure is “/XT99/MXReasonCode/” \*/

**ENDIF**

**IF Length**(MXAdditionalInformation) > 35 – **Length**(MTLine[2]) THEN

MTLine[2] = **Concatenate**(MTLine[2],**Substring**(MXAdditionalInformation,1, 35 – **Length**(MTLine[2]))

NarrativeLeft = **Substring**(MXAdditioanlInformation, 35 – **Length**(MTLine[2])+1)

**ELSE**

MTLine[2] = **Concatenate**(MTLine[2],MXAdditionalInformation)

NarrativeLeft = “”

**ENDIF**

/\* If Narrative left is not empty, copy it to MTLine[6] \*/

**IF Length**(NarrativeLeft) > 0 THEN

/\* Build MTLine[6] \*/

/\* NarrativeLeft is concatenated with MTLine[6], even if total length > 35 \*/

MTLine[6] = **Concatenate**(“/TEXT/”,NarrativeLeft)

**ELSE**

MTLine[6]= “”

**ENDIF**

/\* fill in Field 72 \*/

**For i** = 1 to 5

**IF Lengh**(MTLine[i]) > 0 THEN

**AppendToNextLine**(MTLine[i], MT72)

**ENDIF**

**Next i**

/\* IF Length(MTLine[6])> 0, check if still space to copy in Field 72, possibly using continuation “//” if needed \*/

**IF** **Length**(MTLine[6]) > 0 THEN

/\* Check room left \*/

**IF ReturnFirstLineEmpty** (MT72, 6) = 0 THEN

Flag\_MissingInformation = “True”

#### 

**ELSE**

NumberOfEmptyLines = 6 – **ReturnFirstLineEmpty** (MT72, 6) + 1

/\* Append info to Field 72 using “//” as continuation line indicator \*/

**AppendComplexMT72**(NumberOfEmptyLines, MTLine[6], MT72;MT72)

**ENDIF**

**ENDIF**

**/ \*subfunctions description \*/**

**SubfunbctionReasonCodeTranslation**(MXReasonCode; MTReasonCode)

/\* The table with MT MX reason code equivalence is described in the Excel file pacs.002 to MT199/299 /REJT/, sheet “Error Codes RETN REJT” \*/

/\* In “Error Codes RETN REJT” sheet in EXCEL document, in col H, search for MXReasonCode. Return the corresponding value IsMTErrorCodePresent from column D. If the MXReasonCode is not found in col H (ie., excel list not up to date), IsMTErrorCodePresent = “False”.

If IsMTErrorCodePresent is “True”, extract the MT equivalent code named MTReasonCode from column A \*/

/\* Local variables

MTFixedCode : string

\*/

IsInTable, IsMTErrorcodePresent : look up function returning boolean

MTFixedCode = “/XT99/”

**IF** **IsInTable**(MXReasonCode, col H) THEN

**IF** **IsMTErrorCodePresent**(col D) THEN

MTReasonCode = **GetEquivalent**(MXReasonCode, col A)

MTReasonCode = concatenate(“/”, MTReasonCode, “/”)

**ELSE**

/\* No MT equivalent code \*/

MTReasonCode = **Concatenate**(MTFixedCode,MXReasonCode, “/”)

**ENDIF**

**ELSE**

/\* MX code is not in the table but must still be a valid ISO code. This assumption is to avoid maintenance issue with the table in Excel \*/

MTReasonCode = **Concatenate**(MTFixedCode,MXReasonCode, “/”)

**ENDIF**

**/\* End of SubfunbctionReasonCodeTranslation \*/**

**SubfunctionIDTruncation**(MXID, IsInstructionID; MTID)

/\* This function converts MXID to MTID limited to max 16 char. If not possible to generate a valid MTID value “NOTPROVIDED” is returned \*/

**IF Length**(MXID) = 0 THEN

MTID = “NOTPROVIDED”

**IF** ISInstructionID THEN

T20072

**ELSE**

T20073

**ENDIF**

**ELSE** /\* Length MXID > 0 \*/

**IF Length**(MXID) > 16 THEN

MTID = (**Concatenate**(**Substring**(MXID, 1, 15)),"+"))

/\* Reference truncation should be reported by Error Handling \*/

**ELSE** MTID = MXID

**ENDIF**

**IF** MTID does not comply with '(/.\*)|(.\*/)|(.\*//.\*)'

/\* meaning does not start or end with "/" and does not contain "//" within the string . \*/

THEN return MTID

**ELSE**

MTID = "NOTPROVIDED"

**IF** IsInstructionID = “True” THEN T14002

/\* need a generic reporting message to cover both an invalid OriginalInstructionID or invalid OriginalMsgID \*/

**ELSE**

T14001

**ENDIF**

**ENDIF**

**ENDIF** /\* End IF Length(MXID) = 0 \*/

**/\* End of SubfunctionIDTruncation \*/**

**SubfunbctionChargeCalculation(**MXTransactionInformation;MTLine[5])

/\* Local variables

MXTotalAmount : MX Amount \*/

**For** i = 1 to **NumberOfOccurrences**(MXTransactionInformation.ChargesInformation.Amount)

**IF** MXTransactionInformation.ChargesInformation[i].Amount.XMLAttribute(Ccy) **NOT Equal** to MXTransactionInformation.ChargesInformation[1].Amount.XMLAttribute(Ccy) THEN

T13006 /\* Error code list \*/

MTLine[5] = “”

Exit SubfunctionChargeCalculation

**ELSE**

MXTotalAmount = MXTotalAmount + MXTransactionInformation.ChargesInformation[i].Amount

ENDIF

**Next** i

**IF** MXtotalAmount > 0 THEN

**MX\_To\_MTcurrencyAmount**(MXTotalAmount(with currency), MTCurrency, MTAmount)

**IF** **Length**(MTAmount) > 15 THEN

/\* max 15d in MT, decimal included \*/

Line[5] = “”

T20039 /\*Error code list \*/

**Exit** SubfunctionChargeCalculation

**ENDIF**

Line[5] = **Concatenate**(“/CHGS/”,MTCurrency, MTAmount)

**ELSE**

Line[5] = “”

**ENDIF**

**/\* End SubfunctionChargeCalculation \*/**

**SubfunctionNARRSpecificScenario**(MXAdditionalInformation;IsNARRSpecificScenario, MTReasonCode, MXAdditionalInformation)

/\*LocalVariable

ReasonCode : string \*/

ReasonCode = **Substring**(MXAdditionalInformation,1,4)

ReasonCode = **ExtractPattern**(ReasonCode,[0-9A-Z]{2}[0-9]{2})

/\* this is to check the first 4 char are possible MT codes \*/

**IF Length**(ReasonCode)= 0 THEN

IsNARRSpecificScenario = “False”

Return /\* Exit subfunction \*/

**ELSE**

**IF Withinlist**(ReasonCode, MTReturnCodeList) OR **Length** (**ExtractPattern**(ReasonCode, X[0-9A-Z]{1}[0-9]{2}))> 0 THEN

/\* MTReturnCodeList is defined in excel pacs.002 to MT199 MT299 REJT, spreadsheet “Error Codes RETN REJT”, col A.

Check if Reasoncode is in MT code list or is a bilaterally agreed code \*/

IsNARRSpecificScenario = “True”

MTReasonCode = **Concatenate**(“/”,ReasonCode, “/”)

**IF** **Substring**(MXAdditionalInformation,5,1)= “/” THEN

MXAdditionalInformation = **Substring**(MXAdditionalInformation,6)

/\* remove the MTReasonCode and slash separator \*/

**ELSE**

MXAdditionalInformation = **Substring**(MXAdditionalInformation,5)

/\* this should no happen if the correct structure generated from MT to MX is followed \*/

**ENDIF**

**ELSE**

IsNARRSpecificScenario = “False”

**ENDIF**

**ENDIF**

**/\* End SubfunctionNARRSpecificScenario \*/**

### 4.3.22 MX\_To\_MT72Or79REJT

**Name**

MX\_To\_MT72Or79REJT

**Business description**

The function builds the field 72 used in a reject p ayment (eg MT103 REJT, MT202 REJT) or field 79 in MTn99 when translated from pacs.002 as a source message.

Structure of field 72/79 is as follows:

-Line 1 : /REJT/2!n where 2!n is the tag of the field in error. Default value “99” is used as identification of the tag in error from pacs.002 is not straightforward.

-Line 2 : /2!c2!n/[Parameter\*] contains the MT reject reason code follows by narrative information. The MT code is obtained by conversion from MX code. If no equivalent exists, the default value /XT99/ is used followed by the MX ISO Reason code, ie, /XT99/ISOReasonCode/[Text1]. If the MX Reason is proprietary, it is also added after the code /XT99/MXReasonProprietary/[Text1] where Text1 is the Additional Information. Line 6 is the continuation of Line 2.

-Line 3 : /MREF/16x contains field 20 of the original message (ie payment message)

(ie., in Field 72 => OriginalInstructionID; in Field 79 => OriginalMsgId.)

-Line 4 : /TREF/16x contains the transaction reference (ie OriginalEndToEndID, if value is different rrom “NOTPROVIDED”)

-Line 5 : /CHGS/ is not used as Charges are removed from CBPR+ pacs.002

-Line 6 : /TEXT/Parameter : narrative description that can be continued on next line starting with “//”

In Field 79, the code word /UETR/ follows by the UETR value of the Original Payment is added as last line of the Field.

\*Parameter represents 29x in field 72 and 44x in field 79.

**Format**

MX\_To\_MT72REJT (MXStatusReasonInformation, MXTransactionInformationAndStatus, FieldType; MT72\_79)

**Input**

MXStatusReasonInformation: MX message element typed StatusReasonInformation12

MXTransactionInformationAndStatus: MX message element typed PaymentTransaction110

FieldType has one of the values {72,79}

**Output**

MT72\_79 : field with structure of field 72 (6 lines of 35 characters, see MT103 RETN) OR structure of Field 79 (35 lines of 50 characters, see MTn99) depending on the input parameter fieldType. IF FieldType = “72”, output structure is Field 72 else output structure is Field 79.

**Preconditions**

None

**Formal description**

/\* MX\_To\_MT72Or79REJT is very similar to MX\_To\_MT72RETN despite the fact that the types of input parameters are different**. The subfunctions are described in MX\_To\_MT72RETN** \*/

/\* Local variables

MXReasonCode, MXAdditionalInformation, NarrativeLeft, MTReasonCode, MXReasonProprietary, MTFixedCode, MREF\_ID, UETRCodeWord, MXUETR, MTUETR: string

MTLine[]: table of string

IsInstructionID : Boolean

NumberOfEmptyLines, MaxNumberOfLines, LineLength : integer

\*/

**IF** FieldType = 72 THEN

LineLength = 35

**ELSE**

LineLength = 50

**ENDIF**

MTFixedCode = “/XT99/”

UETRCodeWord = “/UETR/”

/\* UETRCodeWord is used as a parameter, in case it is decided to not translate UETR to Field79, UETRCodeWord would be initiated as an empty string \*/

/\* Extract MX information. With Status “Reject”, the Reason is mandatory \*/

**IF IsPresent**(MXStatusReasonInformation.Reason.Code) THEN

MXReasonCode = MXStatusReasonInformation.Reason.Code

**ELSEIF** **IsPresent** (MXStatusReasonInformation.Reason.Proprietary)

MXReasonProprietary = MXStatusReasonInformation.Reason.Proprietary

**ENDIF**

**IF** **Length**(MXStatusReasonInformation.AdditionalInformation[1]) > 0 THEN

MXAdditionalInformation = MXStatusReasonInformation.AdditionalInformation[1]

**IF** **Length**(MXStatusReasonInformation.AdditionalInformation[2]) > 0 THEN

**IF** **Length**(MXStatusReasonInformation.AdditionalInformation[1]) > 104

/\* Assumption is that the next line is the continuation of the first one and therefore no space is added between the 2 occurrences \*/

THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXStatusReasonInformation.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXStatusReasonInformation.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF**

**/\* Build MTLine 1 \*/**

MTLine[1] = **Concatenate**(“/REJT/”, “99”)

**/\* Build MTLine 3 \*/**

IsInstructionID = “true”

**IF** FieldType = 72 THEN

MREF\_ID = MXTransactionInformationAndStatus.OriginalInstructionID

**ELSE**

MREF\_ID = MXTransactionInformationAndStatus.OriginalGroupInformation/OriginalMessageIdentification)

/\* In MT199/299, OriginalInstructionId is copied to Field 21 optional \*/

**ENDIF**

**Call SubfunctionIDTruncation**(MREF\_ID, IsInstructionID; MTID)

MTLine[3] = **Concatenate**(“/MREF/”, MTID)

**/\* Build MTLine 4 - optional \*/**

**IF Length**(MXTransactionInformationAndStatus.OriginalEndToEndID)> 0 AND

MXTransactionInformationAndStatus.OriginalEndToEndID **NOT Equal** to “NOTPROVIDED” THEN

{ IsInstructionID = “false”

**Call SubfunctionIDTruncation**(MXtransactionInformationAndStatus.OriginalEndToEndID, IsInstructionID; MTID)

MTLine[4] = **Concatenate**(“/TREF/”, MTID)}

**ELSE** MTLine[4] = “”

**ENDIF**

**/\* no MTLine 5 needed \*/**

MTLine[5] **= “”**

**/\* Build MTLine 2 and MTLine 6 \*/**

**IF IsPresent**(MXStatusReasonInformation.Reason.Code)THEN

**Call SubfunbctionReasonCodeTranslation** (MXReasonCode; MTLine[2])

/\* as output of the function MTLine[2] must be “/MTReasonCode/” or “/XT99/MXReasonCode/”

**ELSE**

/\* Reason Proprietary must be present \*/

MTLine[2] = **Concatenate**(MTFixedCode,MXReasonProprietary,”/”)

**ENDIF**

/\* Start to copy MXAdditionalInformation to Line 2 and continue to Line 6. LineLength is an input parameter as described above\*/

**IF Length**(MXAdditionalInformation) > LineLength – **Length**(MTLine[2] THEN

MTLine[2] = **Concatenate**(MTLine[2],**Substring**(MXAdditionalInformation,1, LineLength – **Length**(MTLine[2]))

NarrativeLeft = **Substring**(MXAdditioanlInformation, LineLength – **Length**(MTLine[2])+1)

**ELSE**

MTLine[2] = **Concatenate**(MTLine[2],MXAdditionalInformation)

NarrativeLeft = “”

**ENDIF**

/\* If Narrative left is not empty, copy it to MTLine[6] \*/

**IF Length**(NarrativeLeft) > 0 THEN

/\* Build MTLine[6] \*/

/\* NarrativeLeft is translated to MTLine[6], even if total length > LineLength \*/

MTLine[6] = **Concatenate**(“/TEXT/”,NarrativeLeft)

**ENDIF**

/\* fill in Field 72 or 79 \*/

**For i** = 1 to 5

**IF Lengh**(MTLine[i]) > 0 THEN

**AppendToNextLine**(MTLine[i], MT72\_79)

**ENDIF**

**Next i**

/\* IF Length(MTLine[6])> 0, check if still space to copy in Field 72 or 79, possibly using continuation “//” if needed \*/

**IF** **Length**(MTLine[6]) > 0 THEN

**IF** FieldType = “72” THEN

MaxNumberOfLines = 6

**ELSE**

MaxNumberOfLines = 35

**ENDIF**

/\* Check room left \*/

**IF ReturnFirstLineEmptyExtended** (MT72\_79, MaxNumberOfLines, LineLength) = 0 THEN

Flag\_MissingInformation = “True”

#### 

**ELSE**

NumberOfEmptyLines = MaxNumberOfLines – **ReturnFirstLineEmpty** (MT72\_79, MaxNumberOfLines) + 1

/\* Append info to Field 72 or 79 \*/

**IF** FieldType = “72” THEN

**AppendComplexMT72**(NumberOfEmptyLines, MTLine[6], MT72\_79;MT72\_79)

**ELSE**

**AppendComplexMT79**(NumberOfEmptyLines, MTLine[6], MT72\_79;MT72\_79)

/\* AppendComplexMT79 will do the same as function AppendComplexMT72 but with lines of max 50 char and max number of lines is 35. Due to the similarities between the 2 functions, AppendComplexMT79 is not described in this document \*/

**ENDIF**

**ENDIF**

**ENDIF**

**IF** FieldType = “79” AND **Length**(UETRCodeWord) > 0 THEN

/\* UETRCodeWord is used as a parameter, in case it is decided to not translate UETR to Field79, UETRCodeWord would be initiated as an empty string \*/

MXUETR = MXTransactionInformationAndStatus.OriginalUETR

**IF** **IsEmpty**(MTLine[6]) THEN

MTUETR = Concatenate(“/TEXT/”,UETRCodeWord,MXUETR)

**ELSE**

MTUETR = Concatenate(“//”,UETRCodeWord,MXUETR)

/\* line starts with ///UETR/UETRValue where the first 2 slashes are the continuation slashes in order to comply with field 79 /REJT/ structure \*/

**ENDIF**

/\* Append the lines. No issue with room left as Field 79 allows 35 lines of 50 char. \*/

**AppendToNextLine**(MTUETR, MT72\_79)

**ENDIF**

### 4.3.23 MX\_To\_MTEmptyLine

**Name**

MX\_To\_MTEmptyLine

**Business description**

The function looks in a multiline field (eg Field 70, 77B, 56D **in MT103**) for lines with only CRLF presentor lines with SPACE and CRLF.

Special checks for

-Field 72 with line which cannot be filled only with “//CRLF” or “//SPACE CRLF” in MT103, MT202, MT 205, MT900, MT910. Not relevant in the other MT messages used in CBPR+ translation scope because Field 72 is not present.-Field 50F/59F with line which cannot be filled only with “Number/CRLF” or “Number/SPACE CRLF”.

Those “empty lines” are then removed from the MTMultilineField because they generate an invalid MT

**Format**

MX\_To\_MTEmptyLine (MTMultilineField; MTMultilineField)

**Input**

MTMultilineField : field which as a structure like 70, 72, 77B (in MT103) or Agents with option D Name and Address (eg 56D)

**Output**

Same MTMultilineField but with empty lines removed and replaced by the next lines, if any.

**Preconditions**

**Formal description**

/\* Local variables \*/

SPACE : char containing the “space” char

CRLF : char containing the CRLF char

Line : string

I, Index : integer

IndexTable : table of integer

PatternNumber : 2 char

EmptyLineIndicator : boolean

PatternNumber = “[1-8]/”

/\* Number followed by slash in the structured Address in subfield2 in 50F like “1/”) \*/

Index = 0

/\* Search for invalid empty lines \*/

**For** i = 1 to **NumberOfOccurrences**(MultilineField)

/\* Repeat for each line in MultilineField \*/

Line = MultilineField.Line[i]

/\* Remove CRLF \*/

EmptyLineIndicator = false

**IF** MultilineField.Type = 72 **AND** Line starts with “//”

Line = **TrimRight**(Line, CRLF) Line = **TrimRight**(Line, SPACE)

Line = **TrimLeft**(Line, “//”)

**IF Length**(Line)= 0 THEN

EmptyLineIndicator = true

**ENDIF**

**ELSEIF** MultilineField.Type = Subfield2 in Field 50F OR Subfield2 in Field 59F)

/\* Subfield 2 means Number Name and Address \*/

Line = **TrimRight**(Line, CRLF)

Line = **TrimRight**(Line, SPACE)

**IF** Line = PatternNumber

EmptyLineIndicator = true

**ENDIF**

**ELSE**

/\* Remove lines with SPACES \*/

Line = **TrimRight**(Line, CRLF)

Line = **TrimRight**(Line, SPACE)

**IF Length**(Line)= 0 THEN

EmptyLineIndicator = true

**ENDIF**

**ENDIF**

() **OR** (()

**IF** EmptyLineIndicator = True THEN

/\* store the number of the line which is empty \*/

Index = Index + 1

IndexTable[Index] = i

**ENDIF**

**Next** i

/\* Remove the empty lines from MultilineField which are at the position IndexTable[Index] \*/

**For** i = 1 to **Index**

**DeleteLine(**MultilineField,IndexTable[Index])

~~T20074~~ T0000E /\* Error code decripted in Error code list \*/

**Next** i

### 4.3.24 MX\_To\_MT79CANC

**Name**

MX\_To\_MT79CANC

**Business description**

The function builds the field 79 in a payment cancellation request (eg MT192/MT292) when translated from camt.056 as a source message.

Structure of field 79 is as follows 35\*50x with:

Line 1 : /4!c/[AdditionalInformation]

Line 2-35 //Continuation of Additional Information

Line [Last] : /UETR/UETRNumber

/UETR/UETRNumber is the last line used in Field79 in order to be aligned with gpi structure expecting the cancellation request reason /4!c/ on the first line of field 79.

**Format**

MX\_To\_MT79CANC (MXOriginalUETR, MXCancellationReasonInformation; MT79)

**Input**

MXCancellationReasonInformation: MX message component typed PaymentCancellationReason5

MXOriginalUETR: MX message component typed UUIDv4Identifier (from TransactionInformation.OriginalUETR)

**Output**

MT79 : field with structure of field 79 (35\*50x)

**Preconditions**

None

**Formal description**

/\* Local variables

MXReasonCode, MXAdditionalInformation, 79String, UETRCodeWord, MXUETR : string

\*/

UETRCodeWord = “/UETR/”

/\* Extract MX information \*/

MXUETR = MXOriginalUETR

MXReasonCode = MXCancellationReasonInformation.Reason.Code

**IF** **Length**(MXCancellationReasonInformation.AdditionalInformation[1]) > 0 THEN

MXAdditionalInformation = MXCancellationReasonInformation.AdditionalInformation[1]

**IF** **Length**(MXCancellationReasonInformation.AdditionalInformation[2]) > 0 THEN

**IF** **Length**(MXCancellationReasonInformation.AdditionalInformation[1]) > 104

/\* Assumption is that the next line is the continuation of the first one and therefore no space is added between the 2 occurrences \*/

THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXCancellationReasonInformation.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXCancellationReasonInformation.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF**

/\* Build /4!c/AdditionalInformation \*/

/\* Translate to structure /4!c/[Additional Information]. If next lines are needed they must start with “//” \*/

79String = **Concatenate**(“/”,MXReasonCode,”/’,MXAdditioanlInformation)

**AppendComplexMT79**(35,79String,MT79;MT79)

/\* The function AppendComplexMT79 is similar to AppendComplexMT72 *but fill in lines with max 50 characters*. Due to the similarities, the function is not formally defined as AppendComplexMT72 could be used with a parameter for the length of a line. Each line after the first one starts with “//”. No truncation is expected. \*/

/\* Build last Line with UETR \*/

**IF Length**(UETRCodeWord) > 0 THEN

/\* UETRCodeWord is used as a parameter, in case it is decided to not translate UETR to Field79, UETRCodeWord would be initiated as an empty string \*/

MTUETR = **Concatenate**(UETRCodeWord,MXUETR)

**AppendToNextLine**(MTUETR,MT79)

**ENDIF**

### 4.3.25 MX\_To\_MT76RCANC

**Name**

MX\_To\_MT76RCANC

**Business description**

The function builds the field 76 in MT196/MT296 as a response to a payment cancellation request when translated from camt.029 as a source message.

Structure of field 76 is as follows 6\*35x with:

Line 1 : /4!c/[AdditionalInformation1] [[/]AdditionalInformation2]

Line 2-6 //Continuation of Additional Information

Where

-/4!c/ contains the cancellation status {CNCL, PDCR, RJCR}

-AdditionalInformation1 contains the MXReasonCode, if present

-AdditionalInformation2 contains MXAdditionalInformation, if present. Both are independent. If Status =”RJCR” then MXReasonCode is mandatory.

Field 77A (20\*35x) will be used if no room enough in field 76. MX UETR will be copied in the last 2 lines used of field 77A with code word /UETR/. Continuation slash “//” is used on the second line.

**Format**

MX\_To\_MT76RCANC (MXStatus, MXCancellationStatusReasonInformation, MXUETR; MT76, MT77A)

**Input**

MXStatus: MX message element type CBPR\_CancellationStatus

MXCancellationStatusReasonInformation: MX message component typed CancellationStatusReason4

MXUETR : UUIDv4Identifier (from TransactionInformationAndStatus.OriginalUETR)

**Output**

MT76 : field with structure of field 76 (6\*35x)

MT77A : field with structure of field 77A (20\*35x)

**Preconditions**

None

**Formal description**

/\* Local variables

MXReasonCode, MXAdditionalInformation, MXString76 : string

MTMultiLines : multiLines field with max 26 lines of 35 char

MaxLineNumber, i, NumberOfEmptyLines : integer \*/

/\* Extract information \*/

**IF IsPresent**(MXCancellationStatusReasonInformation.Reason)THEN

MXReasonCode = MXCancellationStatusReasonInformation.Reason.Code

**ENDIF**

**IF** **Length**(MXCancellationStatusReasonInformation.AdditionalInformation[1]) > 0 THEN

MXAdditionalInformation = MXCancellationStatusReasonInformation.AdditionalInformation[1]

**IF** **Length**(MXCancellationStatusReasonInformation.AdditionalInformation[2]) > 0 THEN

**IF** **Length**(MXCancellationStatusReasonInformation.AdditionalInformation[1]) > 104

/\* Assumption is that the next line is the continuation of the first one and therefore no space is added between the 2 occurrences \*/

THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXCancellationStatusReasonInformation.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXCancellationStatusReasonInformation.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF**

/ \*Build the string \*/

MXString76 = Concatenate(“/”, MXStatus,”/”)

/\* MXStatus is passed as input parameter \*/

**IF Length**(MXReasonCode) > 0 THEN

**IF Length**(MXAdditionalInformation) > 0 THEN

MXString76 = **Concatenate**(MXString76,MXReasonCode,”/”,MXAdditionalInformation)

**ELSE**

MXString76 = **Concatenate**(MXString76,MXReasonCode)

**ENDIF**

**ELSE**

**IF Length**(MXAdditionalInformation) > 0 THEN

MXString76 = **Concatenate**(MXString76,MXAdditionalInformation)

**ENDIF**

**ENDIF**

/\* MXString76 will be formatted as a MT multilines field, each line of max 35 char with continuation lines starting with “//”. More than 6 lines might be needed. That is why first the field to be built is an intermediary one named MTMultiLines and then once the MTMultiLines is built it will be split into field 76 and 77A if needed. No truncation expected. \*/

MTMultiLines = **SplitInLines**(MXString76,35,”//”)

/\* Search the number of lines used in MultiLines \*/

MaxLineNumber = **NumberOfOccurrences**(MTMultiLines)

**IF** MaxLineNumber > 0 THEN

**IF** MaxLineNumber < 7 THEN

**For i** = 1 to MaxLineNumber

**AppendToNextLines**(MTMultiLines[i], MT76)

**Next i**

**ELSE**

**For i** = 1 to 6

**AppendToNetxLines**(MTMultiLines[i], MT76)

**Next i**

**For i** = 7 to MaxLineNumber

**AppendToNextLines**(MTMultiLines[i], MT77A)

**Next i**

**ENDIF**

/\* Fill in UETR in last used line of MT77A. As MT77A has 20\*35x there is no truncation issue but 2 lines are needed and the continuation “//” is used on the second line \*/

/\* AppendComplexMT77A will do the same as function AppendComplexMT72 but max number of lines is 20. Due to the similarities between the 2 functions, AppendComplexMT77A is not described in this document \*/

MXUETR = **Concatenate** (“/UETR/”, MXUETR)

**IF** **ReturnFirstLineEmpty** (MT77A, 20) = 0 THEN

Flag\_MissingInformation = “True”

Exit Function MX\_To\_MT76RCANC

**ENDIF**

NumberOfEmptyLines = 20 – **ReturnFirstLineEmpty** (MT77A, 20) + 1

/\* Append info to Field 77A \*/

**AppendComplexMT77A**(NumberOfEmptyLines, MXUETR, MT77A;MT77A)

### 4.3.26 MX\_To\_MT23E\_BIS

**Name**

MX\_To\_MT23E\_BIS

**Business description**

The function is very similar to MX\_To\_MT23E but in MX\_ToMT23E\_BIS InstructionForCreditorAgent.Code is absent. The function scans InstructionForCreditorAgent.InstructionInformation (MXInstruction) to look for codes in the list {/CHQB/, /HOLD/, /PHOB/,/TELB/}.

If there are 2 occurences of InstructionForCreditorAgent and each of them has no Code in InstructionForCreditorAgent.Code then the 2 occurrences of InstructionInformation are concatenated to cater for the case where a code from the list {/CHQB/, /HOLD/, /PHOB/,/TELB/} is split between the 2 occurrences. If the length of the first occurrence is less than 140 characters and the second occurrence does not start with one of the 4 codes, then the textual information starting the second occurrence is assumed being a new information that will be translated to field 72 /ACC/ using another function (SubfunctionInstructionForCreditorAgentAndJP). Otherwise the 2 occurrences are concatenated in one string that will be analysed to search codes from the list {/CHQB/, /HOLD/, /PHOB/,/TELB/}. All textual information linked to code CHQB is translated to field 72 /ACC/ with the mentioned above subfunction.

Although this structure is not expected, it is still foreseen in translation to cater for cases where the payment is originated in MX. If the payment is originated in MT, the pattern expected is covered in MX\_To\_MT23E where InstructionForCreditorAgent.Code should be present as a result of the translation MT to MX when Field 23E is present with CHQB, HOLD, PHOB or TELB.

MT InstructionInformation linked to one of the codes { /HOLD/, /PHOB/,/TELB/} is truncated if length is greater that 30 characters and truncation is indicated with a “+” sign.

**Format**

**MX\_To\_MT23E\_BIS**(InstructionForCreditorAgent[1], InstructionForCreditorAgent[2]; 23E[n])

**Input**

InstructionForCreditorAgent typed InstructionForCreditorAgent1, max 2 occurrences

**Output**

Field23E (multiple occurrences if any)

**Preconditions**

InstructionForCreditorAgent.Code is absent

**Formal description**

/\* Local variables

MxInstruction, RemainingString : string

n, k, j : integer

Instruction[] : table made of

Instruction.Code : 4 characters

Instruction.AdditionalInformation : string

CodeList = {/CHQB/, /HOLD/, /PHOB/,/TELB/}

\*/

j = 0

/\* Define the string to analyse \*/

**IF IsPresent**(InstructionForCreditorAgent[1] AND **IsAbsent**(InstructionForCreditorAgent[1].Code) AND

**IsPresent**(InstructionForCreditorAgent[2] AND **IsAbsent**(InstructionForCreditorAgent[2].Code)

THEN

{**IF Length**(InstructionForCreditorAgent[1].InstructionInformation)<140 **AND NOT WithinList**(Substring(InstructionForCreditorAgent[2].InstructionInformation),1,6), CodeList)THEN

MXInstruction = **Concatenate**(InstructionForCreditorAgent[1].InstructionInformation,”/TempACC/”, InstructionForCreditorAgent[2].InstructionInformation)

**ELSE**

MXInstruction = **Concatenate**(InstructionForCreditorAgent[1].InstructionInformation, InstructionForCreditorAgent[2].InstructionInformation)

**ENDIF**}

**ElSEIF IsPresent**(InstructionForCreditorAgent[1] **AND IsAbsent**(InstructionForCreditorAgent[1].Code)THEN

MXInstruction = InstructionForCreditorAgent[1].InstructionInformation

**ELSEIF**

**IsPresent**(InstructionForCreditorAgent[2] **AND IsAbsent**(InstructionForCreditorAgent[2].Code)THEN

MXInstruction = InstructionForCreditorAgent[2].InstructionInformation

**ELSE**

**EXIT** function

/\* Case to be covered in MX\_To\_MT23E if InstructionForCreditorAgent.Code is present in the occurrence of InstructionForCreditorAgent \*/

**ENDIF**

/\* Analyse MXInstruction to extract codes in List {/CHQB/, /HOLD/, /PHOB/, /TELB/} if present and related information which are translated to field 23E. IF code /CHQB/ occurs in MXInstruction, the following textual information is translated to field 72 /ACC/ with subfunction *SubfunctionInstructionForCreditorAgentAndJP*

IF MXInstruction starts with textual information or if MXInstruction is full textual information without any of the code words listed, , it is also translated with the subfunction from MX\_To\_MT72FullField. The information related to code “/TempACC/” is also translated to Field 72 /ACC/ in the mentioned subfunction \*/

Code[1] = “/CHQB/”

Code[2] = “/HOLD/”

Code[3] = “/PHOB/”

Code[4] = “/TELB/”

Code[5] = “/TempACC/”

RemainingString = MXInstruction

**For k** = 1 to 5

/\* k counts the number of permutations \*/

**IF** **Length**(RemainingString) > 0 THEN

**IF** **IsPresentPattern**(RemainingString, Code[1]) THEN

MXText = **ExtractBetweenPattern**(RemainingString, Code[1],{Code[2], Code[3], Code[4], Code[5]})

j = j + 1

Instruction[j].Code = **Substring**(Code[1],2,Length(Code[1]-2)

Instruction[j].AdditionalInformation = MXText

FoundPatternText = **Concatenate**(Code[1], MXText)

RemainingString = **DeletePattern**(RemainingString, FoundPatternText)

**IF** Code[1] = “/CHQB/” AND Length(MXText) > 0 THEN

/\* Additional information not allowed in 23E with CHQB \*/

Instruction[j].AdditionalInformation = “”

**ENDIF**

**ENDIF** /\* IsPresentPattern \*/

Temp = Code[1]

**For t** = 2 to 4

Code[t-1] = Code[t]

**Next** **t**

Code[4] = Temp

**ENDIF** /\* Length(RemainingString) > 0 \*/

**Next k**

/\* Translate to field 23E \*/

/\* k below counts the number of instances of 23E to be created \*/

k = 0

**IF** j > 0 **THEN**

**For n**=1 to j

**IF** Instruction[n].Code = “TempACC”

/\*Do not create 23E \*/

**ELSE**

{ k = k+1

**IF** Instruction[n].AdditionalInformation **NOT IsEmpty** THEN

**IF** **Length**(Instruction[n].AdditionalInformation) > 30 THEN

Instruction[n].AdditionalInformation = **Concatenate**(**Substring**(Instruction[n].AdditionalInformation,1, 29), “+”)

**ENDIF**

23E[k]=**Concatenate**(Instruction[n].Instruction Code

“/”, Instruction[n].AdditionalInformation)

**ELSE**

23E[k] = Instruction[n].InstructionCode

**ENDIF**

}

**ENDIF** /\* End Instruction[n].Code = “TempACC” \*/

**n=n+1**

**ENDIF** /\* End j > 0 \*/

/\* translation postconditions are defined in excel file to sort out the order of the code as requested in Field 23E \*/

### 4.3.27 MX\_To\_MTPartyNameAndStructuredAddress1

**Name**

MX\_To\_MTPartyNameAndStructuredAddress1

**Business description**

This function is similar to MX\_To\_MTPartyNameAndStructuredAddress but is used to translate to 59F **when LEI is present and MX Country Code is present**. In the case one line is reserved for LEI translation to 3/LEI… leaving only 1 line for the CountryLine 3/ as Numbers 1/, 2/ and 3/ can be repeated but maximum twice for the same number. LEI translation is handled in another function.

The information will be translated following the order:

1/ Name

2/StreetName, BuildingNumber, BuildingName, Floor, PostBox, Room, Department, SubDepartment

3/Country/ TownName, PostCode, Country SubDivision, TownLocationName, DistrictName

If an element is missing there will be no indication it is missing in the MT string (ie., the position in the MT string will be insufficient to identify unambiguously the type of information)

The separator Comma will be used to separate the element in the MT string except between Country and TownName where “/” is used to be aligned with the MT structure for number “3/”

Country and TownName are mandatory in CBPR+ if structured postal address is used.

As LEI is present then the NameLine 1/ is limited to one occurrence, the StreetLine 2/ is limited to 1 occurrence and the CountryLine is limited to 1 occurrence.

If StreetLine is absent, if there is one line with Number “1/” and Name is longer than 33 characters then one additional occurrence of “1/” is filled in. Maximum 3 lines are filled in the MTNameAndAddress (as 1 line is reserved for LEI).

**Format**

**MX\_To\_MTStructuredPartyNameAndStructuredAddress1**(MXParty; MTNameAndAddress )

**Input**

MXParty: the entire structure of the MXParty typed *PartyIdentification135*.

**Output**

MTNameAndAddress : max 3 lines of 35 char

**Preconditions**

None

**Formal description**

/\* Throughout the function, if translation of the source

component is spread over more than one line of the 4\*35x format, a Carriage Return Line Feed (*CRLF*) will be added

between consecutive lines to comply with the format of an MT field with multiple lines \*/

/\* Local variables

i,NextIndex, NumberOfNameOccurrences : integer

Separ : string

Temp : string

MXNumber1, MXNumber2, MXNumber3 :string

MXTable2[], MXTable3[], MTNameAndAddressTable[] : table of string

NumberRemainingLines : Integer

MXNumber2Present : Boolean \*/

MXTable2[1] = MXParty.PostalAddress.StreetName

MXTable2[2] = MXParty.PostalAddress.BuildingNumber

MXTable2[3] = MXParty.PostalAddress.BuildingName

MXTable2[4] = MXParty.PostalAddress.Floor

MXTable2[5] = MXParty.PostalAddress.PostBox

MXTable2[6] = MXParty.PostalAddress.Room

MXTable2[7] = MXParty.PostalAddress.Department

MXTable2[8] = MXParty.PostalAddress.SubDepartment

MXTable3[1] = MXParty.PostalAddress.Country

MXTable3[2] = MXParty.PostalAddress.TownName

MXTable3[3] = MXParty.PostalAddress.PostCode

MXTable3[4] = MXParty.PostalAddress.CountrySubdivision

MXTable3[5] = MXParty.PostalAddress.TownNameLocation

MXTable3[6] = MXParty.PostalAddress.DistrictName

Separ = “,”

MXNumber1 = MXParty.Name

**IF** **Length**(MXNumber1) = 0 THEN

MXNumber1 = “NOTPROVIDED”

~~T20130~~ T12009

/\* This is not expected in pacs because if BIC is absent then Name is mandatory in CBPR+ but possible in camt (eg., camt.054) \*/

**ENDIF**

/\* Fill in MXNumber2 respecting the following order StreetName, BuildingNumber, BuildingName, Floor, PostBox, Room, Department, SubDepartment and using the separator between elements. Missing information is not indicated. \*/

**For i = 1 to 8**

{

**IF** **Length**(MXTable2[i])> 0 THEN

**IF** **Length**(MXNumber2) = 0 THEN

/\* it is the first element found \*/

MXNumber2 = **Concatenate**(MXTable2[i])

**ELSE**

MXNumber2 = **Concatenate**(MXNumber2,Separ, MXTable2[i])

**ENDIF**

**ENDIF**

}

**Next i**

/\* Fill in MXNumber3 respecting the following order Country, TownName, PostCode, CountrySubDivision, TownLocationName, DistrictName and using the separator between elements except between Country and TownName where “/” is used. In CBPR+, if Structured postal address is used, Country and TownName are mandatory \*/

/\* Start with Country and TownName \*/

MXNumber3 = **Concatenate**(MXTable3[1], “/”)

MXNumber3 = **Concatenate**(MXNumber3, MXTable3[2])

/\* Continue with other elements \*/

**For i** = 3 to 6

{**IF** **Length**(MXTable3[i])> 0 THEN

MXNumber3 = Concatenate (MXNumber3,Separ, MXTable3[i])

**ENDIF**

}

**Next i**

/\* Check if MXNumber2 has meaningful information \*/

**IF** **Length**(MXNumber2) > 0 THEN

MXNumber2Present = “true”

**ELSE**

MXNumber2Present = “false”

**ENDIF**

/\* IF MXNumber2 is absent then Name can use 2 lines \*/

**IF MXNumber2Present** = “true” THEN

**IF** **Length**(MXNumber1)> 33 THEN

MXNumber1 = **Concatenate**(**Substring**(MXNumber1), 1, 32), “+”)

**ENDIF**

MTNameAndAddressTable[1]= **Concatenate**(“1/”, MXNumber1)

/\* Fill in MXNumber2 \*/

**IF Length**(MXNumber2)> 33 THEN

MXNumber2 = **Concatenate**(**Substring**(MXNumber2), 1, 32), “+”)

**ENDIF**

MTNameAndAddressTable[2]= **Concatenate**(“2/”, MXNumber2)

NextIndex = 3

**ELSE**

/\* 2 lines can be used by Name \*/

**IF** **Length**(MXNumber1)> 66 THEN

MXNumber1 = **Concatenate**(**Substring**(MXNumber1), 1, 65), “+”)

**ENDIF**

**IF** **Length**(MXNumber1)> 33 THEN

MTNameAndAddressTable[1]= **Concatenate**(“1/”, **Substring**(MXNumber1,1,33)

MTNameAndAddressTable[2]= **Concatenate**(“1/”, **Substring**(MXNumber1,34)

NextIndex = 3

**ELSE**

MTNameAndAddressTable[1]= **Concatenate**(“1/”, MXNumber1)

NextIndex = 2

**ENDIF**

**ENDIF** /\* End IF MXNumber2Present \*/

/\* fill in the country Line in the next available position in the table \*/

**IF** **Length**(MXNumber3)> 33 THEN

MXNumber3 = **Concatenate**(**Substring**(MXNumber3), 1, 32), “+”)

**ENDIF**

MTNameAndAddressTable[NextIndex]= **Concatenate**(“3/”, MXNumber3)

/\* Fill in the MT structure with CRLF between the lines \*/

**For i** = 1 to 3

**IF Length**(MTNameAndAddressTable[i])> 0 THEN

**AppendToNextLine**(MTNameAndAddressTable[i], MTNameAndAddress)

**ELSE**

EXIT loop

**ENDIF**

**Next i**

### 4.3.28 MX\_To\_MT79CANC2

**Name**

MX\_To\_MT79CANC2

**Business description**

The function builds the field 79 in MT 292 when translated from camt.058 as a source message.

Structure of field 79 is as follows 35\*50x with:

Line 1 : /4!c/[AdditionalInformation]

Line 2-35 //Continuation of Additional Information

[Line [Last] : /UETR/UETRNumber]

/UETR/UETRNumber is optional and will be the last line used in Field79 in order to be aligned with gpi structure expecting the cancellation reason /4!c/ on the first line of field 79.

This function is very similar to MX\_To\_MT79CANC but the paths of the MX elements are different and the CancellationReason message block in camt.058 has a different type than CancellationReasonInformation in camt.056, although the structure is the same. Note also that in camt.058, OriginalItem/UETR is optional while mandatory in camt.056.

One difference is that in MXAdditionalInformation a structure like /4!c/AdditionalInformation with MX Reason Code = “NARR” is taken into account. Such pattern could result from a previous MT to MX translation when the MT error code is not in the ISO list of error codes (MT\_To\_MXField79\_2). Then translation back to MT, the MX code “NARR” is removed from the concatenation as it does not bring any value.

Note that MT 292 to camt.058 is out of scope of CBPR+ translation. Although the below function can cater for it.

**Format**

MX\_To\_MT79CANC2 (MXOriginalUETR, MXCancellationReason; MT79)

**Input**

MXCancellationReason: MX message component typed NotificationCancellationReason1

MXOriginalUETR: MX message component typed UUIDv4Identifier (from OriginalNotification/OriginalNotificationReference/OriginalItem/UETR)

**Output**

MT79 : field with structure of field 79 (35\*50x)

**Preconditions**

None

**Formal description**

/\* Local variables

MXReasonCode, MXAdditionalInformation, 79String, UETRCodeWord, MXUETR : string

FromMTIndicator : Boolean

CodePattern : string of 6 characters starting and ending with “/” and in between 4 characters alpha-numeric, with only upper cases, ie in MT, /4!c/ type.

\*/

UETRCodeWord = “/UETR/”

FromMTIndicator = “False”

CodePattern = **Concatenate**(“/”,[0-9,A-Z]{4}, “/”)

/\* Extract MX information \*/

MXUETR = MXOriginalUETR

MXReasonCode = MXCancellationReason.Reason.Code

**IF** **Length**(MXCancellationReason.AdditionalInformation[1]) > 0 THEN

MXAdditionalInformation = MXCancellationReason.AdditionalInformation[1]

**IF** **Length**(MXCancellationReason.AdditionalInformation[2]) > 0 THEN

**IF** **Length**(MXCancellationReason.AdditionalInformation[1]) > 104

/\* Assumption is that the next line is the continuation of the first one and therefore no space is added between the 2 occurrences \*/

THEN

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, MXCancellationReason.AdditionalInformation[2])

**ELSE**

MXAdditionalInformation = **Concatenate**(MXAdditionalInformation, SPACE, MXCancellationReason.AdditionalInformation[2])

**ENDIF**

**ENDIF**

**ENDIF**

/\* Check if structure in MX AdditionalInformation is like “/4!c/AdditionalInformation” from MT to MX previous translation \*/

**IF** MXReasonCode = “NARR” AND **Length**(MXCancellationReason.AdditionalInformation[1]) > 0 AND **IsPresentPattern**(**Substring**(MXCancellationReason.AdditionalInformation[1],1,6), CodePattern)THEN

FromMTIndicator = “True”

**ENDIF**

/\* Build /4!c/AdditionalInformation \*/

/\* Translate to structure /4!c/[Additional Information]. If next lines are needed they must start with “//” \*/

**IF** FromMTIndicator = “True” **THEN**

79String = MXAdditionalInformation

**ELSE**

79String = **Concatenate**(“/”,MXReasonCode,”/’,MXAdditioanlInformation)

**ENDIF**

**AppendComplexMT79**(35,79String,MT79;MT79)

/\* The function AppendComplexMT79 is similar to AppendComplexMT72 *but fill in lines with max 50 characters*. Due to the similarities, the function is not formally defined as AppendComplexMT72 could be used with a parameter for the length of a line. Each line after the first one starts with “//”. No truncation is expected. \*/

/\* Build last Line with UETR \*/

**IF Length**(UETRCodeWord) > 0 **AND Length**(MXUETR) > 0 THEN

/\* UETRCodeWord is used as a parameter, in case it is decided to not translate UETR to Field79, UETRCodeWord would be initiated as an empty string \*/

MTUETR = **Concatenate**(UETRCodeWord,MXUETR)

**AppendToNextLine**(MTUETR,MT79)

**ENDIF**

### 4.3.29 MX\_To\_MT75CANC

**Name**

MX\_To\_MT75CANC

**Business description**

The function builds the field 75 in a cancellation request

Structure of field 75 is built as follows 6\*35x with:

Line 1 : /ISOCodeShortDescription/[AdditionalInformation]

Line 2-35 //Continuation of Additional Information

Use case : the function is used in camt.108 to MT 111 translation. The MX Reason Code cannot be compared with the value /2n/ defined in MT 111 Field 75 because most of the /2n/ codes corresponds to predefined queries which have no equivalence in camt.108. Instead the function copies the ISO code short textual description to Field 75 in order to avoid the introduction of new MX codewords in MT.

**Format**

MX\_To\_MT75CANC (MXCancellationReason; MT75)

**Input**

MXCancellationReason: MX message component typed ChequeCancellationReason1

**Output**

MT75 : field with structure of field 75 (6\*35x)

**Preconditions**

None

**Formal description**

/\* Local variables

MXReasonCode, MXReasonDescription, MXAdditionalInformation, 75String: string

MXReasonTable[] : table of strings made of MX Reason Code and MX Reason Short Description

\*/

MXReasonTable[1].Code = “LOST”

MXReasonTable[1].Description = “ChequeLost”

MXReasonTable[2].Code = “DUPL”

MXReasonTable[2].Description = “DuplicateCheque”

MXReasonTable[3].Code = “FRAD”

MXReasonTable[3].Description = “FraudulentOrigin”

MXReasonTable[4].Code = “NARR”

MXReasonTable[4].Description = “Narrative”

MXReasonTable[5].Code = “CUST”

MXReasonTable[5].Description = “RequestedByCustomer”

MXReasonCode = MXCancellationReason.Reason.Code

MXReasonDescription = “”

MXAdditionalInformation = MXCancellationReason.AdditionalInformation[1]

**For** i=1 to **NumberOfOccurences**(MXReasonTable)

**IF** MXReasonCode = MXReasonTable[i].Code THEN

MXReasonDescription = MXReasonTable[i].Description

Exit loop

**Next i**

/\* Build /ISOCodeShortDescription/AdditionalInformation \*/

/\* Translate to structure /ISOCodeShortDescription/[Additional Information]. If next lines are needed they must start with “//” \*/

75String = **Concatenate**(“/”,MXReasonDescription,”/’,MXAdditioanlInformation)

**AppendComplexMT72**(6,75String,MT75;MT75)

/\* The function AppendComplexMT72 is used as the format of Field 72 is the same as the built structure of Field 75 \*/

### 4.3.30 MX\_To\_MT76CANC

**Name**

MX\_To\_MT76CANC

**Business description**

The function builds the field 76 in a cancellation status

Structure of field 76 is built as follows 6\*35x with:

Line 1 : /ISOCodeShortDescription/[AdditionalInformation]

Line 2-35 //Continuation of Additional Information

Use case : the function is used in camt.109 to MT 112 translation. The MX Status Code cannot be compared with the value /2n/ defined in MT 112 Field 76 because most of the /2n/ codes corresponds to predefined answers which have no equivalence in camt.109. Instead the function copies the ISO status code short textual description to Field 76 in order to avoid the introduction of the new MX codewords in MT.

Function logic is the same as MX\_To\_MT75CANC

**Format**

MX\_To\_MT76CANC (MXCancellationStatus; MT76)

**Input**

MXCancellationStatus: MX message component typed ChequeCancellationStatus1

**Output**

MT76 : field with structure of field 76 (6\*35x)

**Preconditions**

None

**Formal description**

/\* Local variables

MXStatusCode, MXStatusDescription, MXAdditionalInformation, 76String: string

MXStatusTable[] : table of strings made of MX Status Code and MX Status Short Description

\*/

MXStatusTable[1].Code = “ACCP”

MXStatusTable[1].Description = “Accepted”

MXStatusTable[2].Code = “REJT”

MXStatusTable[2].Description = “Rejected”

MXStatusCode = MXCancellationStatus.Status.Code

MXStatusDescription = “”

MXAdditionalInformation = MXCancellationStatus.AdditionalInformation[1]

**For** i=1 to **NumberOfOccurences**(MXStatusTable)

**IF** MXStatusCode = MXStatusTable[i].Code THEN

MXStatusDescription = MXStatusTable[i].Description

Exit loop

**Next i**

/\* Build /ISOCodeShortDescription/AdditionalInformation \*/

/\* Translate to structure /ISOCodeShortDescription/[Additional Information]. If next lines are needed they must start with “//” \*/

76String = **Concatenate**(“/”,MXStatusDescription,”/”,MXAdditioanlInformation)

**AppendComplexMT72**(6,76String,MT76;MT76)

/\* The function AppendComplexMT72 is used as the format of Field 72 is the same as the built structure of Field 76 \*/

# 5 Annex

**Annex: MT and ISO 20022 “Externalised” MX Clearing System Lists**

|  |  |  |  |
| --- | --- | --- | --- |
| **Country** | **Code Name** | **MTClearing SystemList** | **MXClearing SystemList** |
| Australia | Australian Bank State Branch Code (BSB) | **AU** | **AUBSB** |
| Austria | Austrian Bankleitzahl | **AT** | **ATBLZ** |
| Canada | Canadian Payments Association Payment Routing Number | **CC** | **CACPA** |
| China | Bank Branch code used in China | **CN** | **CNAPS** |
| Germany | German Bankleitzahl | **BL** | **DEBLZ** |
| Greece | Helenic Bank Identification Code | **GR** | **GRBIC** |
| Hong Kong | Hong Kong Bank Code | **HK** | **HKNCC** |
| India | Indian Financial System Code | **IN** | **INFSC** |
| Ireland | Irish National Clearing Code | **IE** | **IENCC** |
| Italy | Italian Domestic Identification Code | **IT** | **ITNCC** |
| Japan | Japan Zengin Clearing Code | **JP** | **JPZGN** |
| New Zealand | New Zealand National Clearing Code | **NZ** | **NZNCC** |
| Poland | Polish National Clearing Code | **PL** | **PLKNR** |
| Portugal | Portuguese National Clearing Code | **PT** | **PTNCC** |
| Russia | Russian Central Bank Identification Code | **RU** | **RUCBC** |
| South Africa | South African National Clearing Code | **ZA** | **ZANCC** |
| Spain | Spanish Domestic Interbanking Code | **ES** | **ESNCC** |
| Switzerland | Swiss Clearing Code (BC Code) | **SW** | **CHBCC** |
| Switzerland | Swiss Clearing Code (SIC Code) | **SW** | **CHSIC** |
| Taiwan | Financial Institution Code | **TW** | **TWNCC** |
| UK | UK Domestic Sort Code | **SC** | **GBDSC** |
| US | CHIPS Participant Identifier | **CP** | **USPID** |
| US | United States Routing Number | **FW** | **USABA** |

# 6 Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Change description | Impacted section(s) | Reference |
| 13 July 2020 | Change the priority order of the elements: CountrySubdivision is followed by TownLocationName | 4.1.9; 4.2.9 | STDMAPS-1046 |
|  | IsMTClearingSystemCodeInList : returned parameter is boolean | 3.2.3 | STDMAPS-1064 |
|  | Replace /RRIN/ by /RRC/ to align with pacs.004 code word | 4.3.20 | STDMAPS-832 |
|  | Add a path variable to shorten the notation | 3.2.5 |  |
|  | Change the definition of ExtractLines. | 2 | STDMAPS-723 |
|  | Align translation LocalInstrument with Excel | 4.3.10 |  |
|  | Typo :IF Length(MTCode8) > 35 … | 4.1.3 | STDMAPS-1121 |
|  | Update function to align with MX\_To\_MTFATFNameAndAddress | 4.1.4 | STDMAPS-1116 |
| 11 May 2021 | Update MX\_To\_MT72 FullField2 to add DebtorAgent translation | 4.3.14 | STDMAPS-1027 |
|  | Update MX\_To\_MT79FullField business description section | 4.3.20 | STDMAPS-1142 |
|  | Update function MT70ROC\_To\_MX35Text  M | 3.3.8 | STDMAPS-1144 |
|  | Update the description of function IsIBAN | 2.0 | STDMAPS-1218 |
|  | Update MT\_To\_MXPartyAccount to translate MT IBAN account to MX IBAN element | 3.1.2 | STDMAPS\_1219 |
|  | Update MT\_To\_MXFinancialInstitutionAccount to translate MT IBAN to MXIBAN element | 3.2.7 | STDMAPS-1219 |
|  | New IsAccount function | 2 | STDMAPS-1242 |
|  | New MT\_To\_MX\_Serial\_CoverScenario | 3.3.19 | STDMAPS-1243 |
|  | Update MX\_To\_MT53A function description | 4.2.11 | STDMAPS-1244 |
|  | Update MX\_To\_MT54A function description | 4.2.10 | STDMAPS-1245 |
|  | Update function description MX\_To\_MT72FullField by replacing /FIN54/ by /FIN53/. No impact on the function itself. The subfunction “SubfunctionInstructionforNextAgent” extracts /FIN53/ and /FIN54/. Currently /FIN54/ is not used anymore but keep it in case of further need. | 4.3.10 | STDMAPS-1246 |
|  | Update function MT\_To\_MXFATFNameAndAddress to handle the case where Issuer is absent in line “6/” and to align with the translation of PartyIdentifier with “CUST” code in the function MT\_To\_MXFATFIdentification. | 3.1.5 | STDMAPS-1226 |
|  | Update the description of the function MX\_To\_MT72FullField2, no impact on the function itself. The subfunction “SubfunctionInstructionforNextAgent” extracts /FIN53/ and /FIN54/. Currently /FIN54/ is not used anymore but keep it in case of further need. | 4.3.14 | STDMAPS-1257 |
|  | Update MX\_To\_MTRemittanceInformation, business description and Case 3. /URI/ is not used if none of the codeword in the list {/ULTB/, /ULTD/, /PURP/, /ROC/}is present. | 4.3.8 | STDMAPS-1290 |
|  | Update MT\_To\_MXFATFNameAndAddress section translating “7/”. Made more explicit the extraction of NIDN Number to cater cases where only the country code is provided | 3.1.5 | STDMAPS-1340 |
|  | Update MT\_To\_MXPartyNameAndAddress: remove dummy value “NOTPROVIDED” in MX AddressLine | 3.1.6 | STDMAPS-1300 |
|  | Update MX\_To\_MTFinancialInstitutionNameAndUnstructuredAddress. Remove translation of AddressLine if value is “NOTPROVIDED” (ie avoid to have subfield 2 in option D with value “NOTPROVIDED” after the Name) | 4.2.8 | STDMAPS-1349 |
|  | Update MX\_To\_MTPartyNameAndUnstructuredAddress in order to not translate MX Address Line when value is “NOTPROVIDED” | 4.1.8 | STDMAPS-1367 |
|  | Add T20075 and T20076 in MX\_To\_MTClearingIdentifier in order to explain that in some cases ClearingSystemMemberID is not translated to MT as it may generate conflict with BIC in case of discrepancy and then is not allowed in MT with BIC. | 4.2.2 | STDMAPS-1408 |
|  | Rename all error codes replacing “T0…” by “T2…” | All | STDMAPS-1241 |
|  | Add T20075 and T20077 in MX\_To\_MTClearingIdentifierAndChannel | 4.2.3 | STDMAPS-1408 |
|  | Update the function ExtractLines in Basic functions table | 2 | STDMAPS-1422 |
|  | Update function MT\_To\_MXField72NewCodeWords to translate correctly elements to ISO code elements instead of translating to Proprietary | 3.3.18 | STDMAPS-1435 |
|  | Add warning T20086 in MT\_To\_MXFinancialInstitutionAccount | 3.2.7 | STDMAPS-1447 |
|  | Update function MX\_To\_MTFATFIdentification | 4.1.2 | STDMAPS-1473 |
|  | Update MX\_To\_MTFATFID\_CUST\_NIDN | 4.1.5 | STDMAPS-1474 |
|  | Update MT\_To\_MXAccount35 | 3.1.3 | STDMAPS-1505 |
|  | Update subfunction SubfunctionInstructionForNextAgent in MX\_To\_MT72FullField2, add SubfunctionRemoveBIC | 4.3.14 | STDMAPS-1510 |
| 25 July 2021 |  |  |  |
|  | Amend MT\_To\_MXUltimateParty. If TownName is absent for UltimateDebtor, dummy value “NOTPROVIDED” is used to still allow translation of the Country | 3.1.8 | STDMAPS-1515 |
|  | Improve SubfunctionInstructionForNextAgent | 4.3.14 | STDMAPS-1510 |
|  | Amend function IsMTClearingsystemCodeinList to support the case where the payment is originated in MX and the MX ClearingSystemIdentification (5 char) has no MT equivalent (2 char), the MX ClearingSystemIdentification (5 char) has been copied as such in the MTPartyIdentifier after concatenation with the MemberID. So translation back to MX is possible. | 3.2.3 | STDMAPS-1542 |
|  | Amend function MT\_To\_MXClearingIdentifier : same rationale as above | 3.2.2 | STDMAPS-1543 |
|  | Amend function MX\_To\_MTUltimateParty to translate BIC | 4.1.6 | STDMAPS-1546 |
|  | Amend the textual description in MX\_To\_MTRemittanceInformation to add BIC translation | 4.3.8 | STDMAPS-1546 |
|  | Amend the textual description in MX\_To\_MTRemittanceInformation2 to add BIC translation | 4.3.17 | STDMAPS-1546 |
|  | Update function MT\_To\_MXUltimateParty and subfunction SubfunctionExtractInformation to translate back BIC to MX as consequence of CR STDMAPS-1546 | 3.1.8 | STDMAPS-1548 |
|  | Update SubfunctionInstructionForCreditorAgentAndJP | 4.3.10 | STDMAPS-1562 |
|  | Add T20130 in MX\_To\_MTFATFNameAndAddress2 | 4.1.4 | STDMAPS-1563 |
|  | Add T20130 in MX\_To\_MTPartyNameAndUntructuredAddress | 4.1.8 | STDMAPS-1563 |
|  | Add T20130 in MX\_To\_MTPartyNameAndStructuredAddress and update comment related to T20130 | 4.1.9 | STDMAPS-1563 |
|  | Update MX\_To\_MTFATFNameAndAddress to allow Name to be absent (which is possible in camt, eg camt.054), update the business description and formal description | 4.1.3 | STDMAPS-1563 |
|  | Add warning T20131 in MX\_To\_MT23E | 4.3.12 | STDMAPS-1553 |
|  | Update business description in MX\_To\_MT23E | 4.3.12 | STDMAPS-1567 |
|  | Rename warning T20130 into T20182 in section 4.1.4, 4.1.8, 4.1.9  Rename T20131 into T20183 in section 4.3.12 |  | STDMAPS-1568 |
|  | Update MX\_To\_MTClearingIdentifierAndChannel to group the checks on BIC (first US BIC then other BICS) | 4.2.3 | STDMAPS-1569 |
|  | Update function MX\_To\_MTClearingIdentifier | 4.2.2 | MAPLIB-237 |
|  | Amend SubfunctionInstructionForNextAgent to explain more precisely how to extract the code word /FIN53/ | 4.3.14 | MAPLIB-239 |
|  | In MX\_To\_MT72FullField2, in the business description, replace /PHONEBEN/ by /PHONBEN/ and in the subfunction SubfunctionInstructionForCreditorAgent, in a comment replace /PHONEBEN/ by /PHONBEN/. | 4.3.14 | MAPLIB-258 |
|  | Upgrade MT\_To\_MXFATFIdentification to cater for translation back to OrgID Other ID | 4.1.4 | MAPLIB-284 |
|  | Amend MT\_To\_MXFATFNameAndAddress in order to reinforce the conditions under which LEI is translated to MX. | 3.1.5 | MAPLIB-259 |
| 27 October 2021 | Update the function MX\_To\_MTAgent to limit the translation to 2 lines in | 4.2.6 | MAPLIB-288 |
|  | In SubfunctionIntermediaryAgents add an input parameter in the function MX\_TO\_MTAgent call (Intermediary2 and Intermediary3) | 4.3.14 | MAPLIB-288 |
|  | Improve SubfunctionInstructionforCreditorAgentAndJP to also translate information related to /CHQB/ in InstructionInformation when a code other then /CHQB/ is present in InstructionForCreditorAgent.Code | 4.3.10 | MAPLIB-335 |
|  | Remove truncation warning in MX\_To\_MT23E as a consequence of MAPLIB-335. Amend textual descriptions. | 4.3.12 | MAPLIB-336 |
|  | Update SubfunctionInstructionForNextAgent: remove /FIN53/BIC only if SettlementMethod is INGA or INDA  Add input parameter MXSettlementMethod in MX\_To\_MT72FullField and MX\_To\_MT72FullField2 | 4.3.14 | MAPLIB-358 |
|  | Update MX\_To\_MT23E\_BIS : improve the concatenation of the 2 occurrences | 4.3.26 | MAPLIB-388 |
|  | Rewrite SubfunctionInstructionForCreditorAgentAndJP | 4.3.10 | MAPLIB-388 |
|  | Amend MT\_To\_MXIntermediaryAgent: check if ClearingSystem is in ISO list instead of using weaker pattern. | 3.2.8 | MAPLIB-446 |
|  | MX\_To\_MT53A is complementary to SubfunctionInstructionForNextAgent. So the way the 4 occurrences of InstructionForNextAgent are concatenated to extract code words must be the same (impact when the code words and related info are split between occurences.  Remove T20027 as /FIN53/ is translated in SubfunctionInstructionForNextAgent | 4.2.11 | MAPLIB-447 |
|  | Reinforce the identification in MX of scenario where ClearingSystemCode in MT has no MX equivalent and is translated to Name And AddressLine[1]when translation back to MT is needed. | 4.2.12 | MAPLIB-334 |
|  | Amend MX\_To\_MTNameAndAddressToClearingSystemIdentifier | 4.2.5 | MAPLIB-473 |
|  | Amend MT\_To\_MXClearingSystemToNameAndAddressLine | 3.2.4 | MAPLIB-475 |
|  | Amend MX\_To\_MTAgentGeneric : when MT ClearingSystemMemberId is copied to name, up to 35 char are allowed as “//” in included (as per MAPLIB-475) | 4.2.12 | MAPLIB-568 |
|  | Only one table for MT MX clearing conversion in section 5. The table in section 3.2.2 and in section 4.2.2 are removed. TW/TWNCC was missing in section 4.2.2 | 5 | MAPLIB-596 |
|  | Amend MX\_To\_MTClearingIdentifierAndChannel. Error T20075 and T20077 are triggered only if there is a MX ClearingSystemMemberID. Otherwise, errors are not relevant. | 4.2.3 | MAPLIB-601 |
|  | Amend MX\_To\_MTAddressLineType to reinforce the pattern extracting the country code | 4.3.15 | MAPLIB-632 |
|  | Amend MX\_To\_MTFATFIdentification to handle the translation of new ISO codes in PrivateID/OtherID/SchemeName/Code (like TELE) to MT with code “CUST”. | 4.1.2 | MAPLIB-651 |
|  | Amend MX\_To\_MTFATFIdentification to allow the translation of OrganisationID/Other/ID only if OrganisationID/../SchemeName/Code is present in the ISO list. | 4.1.2 | MAPLIB-652 |
|  |  |  |  |
| 1 February 2022 | Clarify the definition of ROCPattern in MX\_TO\_MTRemittanceInformation | 4.3.8 | MAPLIB-680 |
|  | Only the first occurrence of OrgID/Other or PrivateID/Other is translated in MX\_To\_MTUltimateParty | 4.1.6 | MAPLIB-682 |
|  | Update SubfunctionInstructionForNextAgent to support the scenario where /FIN53/ is present and SettlementAccount is absent. Add Business Application Header as input parameter in MX\_To\_MT72FullField and in MX\_To\_MT72FullField2 | 4.3.10  4.3.14 | MAPLIB-700 |
|  | CR on UG v2.1. InstructionForNextAgent format is updated to 6 occurrences of max 35 char each of them. This impacts SubfunctionInstructionForNextAgent and MX\_To\_MT53A | 4.3.14  4.2.11 | MAPLIB-708 |
|  | Update MX\_To\_MTAgentGeneric : update the criteria to call SubfunctionOnlyClearingSystemMemberID and SubfunctionOnlyAccount | 4.2.12 | MAPLIB-742 |
|  | Amend MX\_To\_MTFATFIdentification to cater for new codes in PrivateID/SchemeName/Code | 4.1.2 | MAPLIB-765 |
|  | IsIBAN function adds the check on the pattern as defined in the UGs. | 2 | MAPLIB-767 |
|  | Redesign SubfunctionInstructionForCreditorAgent to improve extraction of codes in InstructionInformation and to extract the new code /UDLC/ requested in pacs.009 CORE when linked to pacs.009 ADV. | 4.3.14 | MAPLIB-784 |
|  | Update description of MX\_To\_MTRemittanceInformation2 to define UltimateCreditor is translated before UltimateDebtor. | 4.3.17 | MAPLIB\_787 |
|  | Amend MT\_To\_MXInstructionForCreditorAgent2 to cater for code /UDLC/ to be translated back to InstructionForCreditorAgent. | 3.3.16 | MAPLIB-795 |
|  | Amend MX\_To\_MTEmptyLine | 4.3.23 | MAPLIB-803 |
|  | Alignement error/warning codes with Translation portal implementation.  Replace T20033 by T11002 in MT\_To\_MXClearingSystemToNameAndAddressLine  Replace T20034 by T11003 in MT\_To\_MXClearingSystemToNameAndAddressLine  Replace T20030 by T11004 in MT\_To\_MXIntermediaryAgent  Replace T20075 by T11005 in MX\_To\_MTClearingIdentifier  Replace T20075 by T11005 in MX\_To\_MTClearingIdentifierAndChannel  Replace T20076 by T11006 in MX\_To\_MTClearingIdentifier  Replace T20077 by T11007 in MX\_To\_MTClearingIdentifierAndChannel  Replace T20086 by T11008 in MT\_To\_MXFinancialInstitutionAccount  Replace T20037 by T12005 in MX\_To\_MTFATFID\_CUST\_NIDN  Replace T20058 by T12001 in MX\_To\_MTFATFIdentification  Replace T20059 by T12002 in MX\_To\_MTFATFIdentification  Replace T20060 by T12003 in MX\_To\_MTFATFIdentification  Replace T20026 by T12004 in MX\_To\_MTFATFIdentification  Replace T20079 by T12006 in MT\_To\_MXUltimateParty  Replace T20182 by T12009 in MX\_To\_MTFATFNameAndAddress2  Replace T20182 by T12009 in MX\_To\_MTPartyNameAndUnstructuredAddress  Replace T20130 by T12009 in MX\_To\_MTPartyNameAndStructuredAddress  Replace T20217 by T12010 in MX\_To\_MTFATFIdentification  Replace T20035 by T14001 in MX\_To\_MT72RETN  Replace T22003 by T13006 in MX\_To\_MT72RETN SubfunctionChargeCalculation  Replace T20095 by T14002 in MX\_To\_MT72RETN SubfunctIDTruncation | 3.2.4  3.2.8  4.2.2  4.2.3  3.2.7  4.1.5  4.1.2  3.1.8  4.1.4  4.1.8  4.1.9  4.3.21 | N/A |
| August 2022 | The function MT\_To\_MXFinancialInstitutionAccount has been updated to handle properly the case where only "/C" or /D" is provided without any account number. | 3.2.7 | MAPLIB-910 |
|  | MX\_To\_MT76RCANC has been amended. Code word /UETR/is added to Field 77A to translate the UETR. | 4.3.25 | MAPLIB-925 |
|  | In MT\_To\_MXField76, amend section where it is analysed if field 76 is structured or unstructured. | 3.3.26 | STDSQA\_680 |
|  | Add a comment in the business description in MT-To\_MXField79 | 3.3.25 | N/A |
|  | Amend MX\_To\_MTAgentGeneric, section calling SubfunctionExceptionCase, AddreLine[2] must be absent | 4.2.12 | STDSQA-699 |
|  | Amend MX\_To\_MTPartyNameAndStructuredAddress, Subfunction Case12. Need to limit “3/” to max 2 occurrences. | 4.1.9 | STDSQA-702 |
|  | Amend MX\_To\_MTAddressLineType : check if the number of lines starting with “2/” does not exceed 2 lines and if the number of lines starting with “3/” does not exceed 2 lines. | 4.3.15 | STDSQA-706 |
|  | Amend MX\_To\_MTPartyNameAndAddressLEI1 to take into account that “3/” can only be present maximum 2 times in 59F. | 4.1.11 | SDTSQA-709 |
|  | Amend MX\_To\_MTPartyNameAndAddressLEI2 to take into account that “3/” can only be present maximum 2 times in 59F. | 4.1.10 | STDSQA-710 |
|  | Amend MX\_To\_MTFATFIdentification to allow extraction of the country code from the MXAddressLine if the MXAddressLine structure is compliant with 50F NameAndAddress structure. This is just in case the Country cannot be found in MX Issuer as it should be if MX results from MT to MX translation. | 4.1.2 | MAPLIB-1009 |
|  |  |  |  |
|  | In MX\_To\_MT72FullField, update criteria for calling SubfunctionCategoryPurpose in order to still allow translation of values “INTC” or “CORT” when misused in Proprietary while it should be used in Code. In order not to loose data, it is translated with /CAPTPURP/ and not to 23E due to misuse. | 4.3.10 | MAPLIB-1060 |
|  | Amend MX\_To\_MT76RCANC to copy UETR on 2 lines with the continuation “//” on the second line | 4.3.25 | STDSQA-739 |
|  | Add comment in the business description about LEI in 50F in MT\_To\_MXPartyNameAndStructuredAddress | 3.1.7 | N/A |
|  | Update the business description in MT\_To\_MXIntermediaryAgent and add check on length of elements to be translated to MX Name and MX TownName | 3.2.8 | STDSQA-781 |
|  | In MX\_To\_MTCurencyAmount, remove leading and trailing spaces in MX amount if any. | 4.3.7 | N/A |
|  | Update MX\_To\_MTPartyNameAndaddressLEI2. Replace at 2 locations  MTNameMTNameAndAddressTable [2] = Concatenate(“1/”, MXName) by  MTNameMTNameAndAddressTable [2] = Concatenate(“1/”, Substring(MXName,34)) | 4.1.10 | STDSQA-806 |
|  | Amend MX\_To\_MTFATFID\_CUST\_NIDN to allow extraction of Country code from Structured AddressLine | 4.1.5 | STDSQA-841 |
|  | Amend MX\_To\_MTAgentGeneric. The pseudo-code calling SubfunctionExceptionCase is amended as well as SubfunctionExceptionCase to improve translation back to option B with Location and/or “//RT” and Option C with “//RT” resulting from previous translation MT to MX. | 4.2.12 | STDSQA-844 |
|  | Amend description of MT\_To\_MXField76 by replacing “Lines 2-35” by “Lines 2-NumberOfLines” as maximum number of lines for Field 76 is 6. No impact on the pseudo code. | 3.3.26 | N/A |
| June 2023 |  |  |  |
|  | Typo. Replace “MTInstructionForNextAgent” by “MTInstruction” in MX\_To\_MT53A | 4.2.11 | N/A |
|  | Improve the concatenaton in MT\_To\_MXField79 by removing unexpected spaces | 3.3.25 | STDSQA-855 |
|  | Improve the concatenaton in MT\_To\_MXField76 by removing unexpected spaces | 3.3.26 | STDSQA-857 |
|  | If a 4th occurrence of /INS/ is present in field 72, it is not translated. Improve MT72INS\_To\_MXAgent to report that information is ignored in translation | 3.2.6 | STDSQA-897 |
|  | Add Assumption in MT72INS\_To\_MXAgent : MX Agent only allows FI BIC. So the BIC following /INS/ must be a FI BIC. Using a non FI BIC after /INS/ will generate an invalid MX message. | 3.2.6 | N/A |
|  | Replace T20050 by T0000T in MT\_To\_MXAccount35 | 3.1.3 | STDSQA-910 |
|  | Replace T20074 by T0000E in MX\_To\_MTEmptyLine | 4.3.23 | STDSQA-910 |
|  | Improve MT\_To\_MXFATFIdentification to translate back Private ID Scheme Name code correctly when the code has no MT equivalent (e.g. POID, TELE) | 3.1.4 | STDSQA-987 |
|  | In MT\_To\_MXFinancialInstitutionAccount, replace T20233 by T11013 | 3.2.7 | STDSQA-889 |
|  | Amend MT\_To\_MXPartyAccount to handle properly the case where account is “//CH” without any following characters | 3.1.2 | MAPLIB-1754 |
|  | Amend MT\_To\_MXPartyNameAndStructuredAddress. In the section searching for “3/”, the first “next I” has to be removed. Probably correctly implemented. | 3.1.7 | STDSQA-1066 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| To Read | Please note that all the above listed items will not necessary be part of the release 1.5.2 but will be in one of the next releases. Refer to the Release Note for more information. |  |  |

# Legal Notices

Copyright

SWIFT © 2022. All rights reserved.

Restricted Distribution

Do not distribute this publication outside your organisation unless your subscription or order expressly grants you that right, in which case ensure you comply with any other applicable conditions.

Disclaimer

The information in this publication may change from time to time. You must always refer to the latest available version.

SWIFT Standards Intellectual Property Rights (IPR) Policy - End-User License Agreement

SWIFT Translation Rules are licensed subject to the terms and conditions of the *SWIFT Translation Rules IPR Policy - End-User License Agreement*, available at [www.swift.com > About Us > Legal > IPR Policies > SWIFT Translation Rules IPR Policy.](https://www.swift.com/about-us/legal/ipr-policies/translation-ipr-policy?tl=en#topic-tabs-menu)

Translations

The English version of SWIFT documentation is the only official and binding version

Trademarks

SWIFT is the trade name of S.W.I.F.T. SC. The following are registered trademarks of SWIFT: 3SKey, Innotribe, MyStandards, Sibos, SWIFT, SWIFTNet, SWIFT Institute, the Standards Forum logo, the SWIFT logo, SWIFT gpi with logo, the SWIFT gpi logo, and UETR. Other product, service, or company names in this publication are trade names, trademarks, or registered trademarks of their respective owners.